2021

Distributed, Scalable And Resilient Information Acquisition For Multi-Robot Teams

Brent Schlotfeldt
University of Pennsylvania

Distributed, Scalable And Resilient Information Acquisition For Multi-Robot Teams

Abstract

Advances in robotic mobility and sensing technology have the potential to provide newcapabilities in a wide variety of information acquisition problems including environmental monitoring, structure inspection, localization and mapping of unknown environments, and search and rescue, amongst many others. In particular, teams composed of multiple robots have shown great potential in solving these problems, though it is challenging to design efficient algorithms that are distributed and scale well, and even more complex in hazardous or challenging environments. The purpose of this dissertation is to provide novel algorithms to the capabilities of multi-robot teams to gather information which are distributed, scalable, and resilient. The first part of the dissertation introduces the single-robot information acquisition problem, and focuses on algorithms that may be used for individual robots to plan their own trajectories. The methods presented here are search-based, meaning that an individual robot has a finite set of actions and is seeking to efficiently build a search tree over a known planning horizon. The first method presented details how to use the concept of algebraic redundancy and closeness to achieve a smooth trade-off of completeness in the exploration process, as an anytime planning algorithm. Next we show how a single robot can compute an admissible and consistent heuristic which guides the search towards the most informative regions of the state space, using the classic A* planning algorithm, drastically improving the search efficiency. The next chapter of the dissertation focuses on how to build on the single robot planning algorithms to create efficient algorithms for multi-robot teams, which operate in a distributed manner and scalable manner. The first method presented is coordinate descent, 5 otherwise known in the literature as sequential greedy assignment. This algorithm is implemented in a multi-robot target tracking hardware experiment. Next, we formulate an energy-aware multi-robot information acquisition problem, which allows for heterogeneity and captures trade-offs between information and energy expenditure. However, this results in a non-monotone objective function. Therefore we propose a new algorithm based on distributed local search, which achieves performance guarantees through a diminishing returns property known as submodularity. The final chapter focuses on hazardous or failure prone environments that necessitate resilience to a fixed number of failures in the multi-robot team. We provide a definition of resilience, and formulate a resilient information acquisition problem. We then propose the first algorithm that solves this problem through an online application of robust trajectory planning, and provide theoretical guarantees on its performance. We then present three unique applications of the resilient multi-robot information acquisition framework, including target tracking, occupancy grid mapping, and persistent surveillance which demonstrate the efficacy of our approach.

Degree Type

Dissertation

Degree Name

Doctor of Philosophy (PhD)

First Advisor

George J. Pappas

Keywords

Control, Engineering, Information Gathering, Multi-Agent, Robotics

DISTRIBUTED, SCALABLE AND RESILIENT INFORMATION ACQUISITION FOR MULTI-ROBOT TEAMS

Brent Thomas Schlotfeldt

A Dissertation

in

Electrical and Systems Engineering

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

2021

Supervisor of Dissertation

George Pappas, Professor of Electrical and Systems Engineering

Graduate Group Chairperson

Victor Preciado, Associate Professor of Electrical and Systems Engineering

Dissertation Committee

Vijay Kumar, Nemirovsky Family Dean of Penn Engineering and Professor of Mechanical Engineering and Applied Mechanics
Hamed Hassani Assistant Professor of Electrical and Systems Engineering
Nikolay Atanasov Assistant Professor of Electrical and Computer Engineering, University of California San Diego

Distributed, Scalable and Resilient Information Acquisition for Multi-Robot Teams

To Kate, who is always inspiring me to be better

Acknowledgments

I am most grateful to my advisor, George Pappas for choosing me as a graduate student, and for providing me with plentiful research opportunities and all the resources I needed to be successful in this journey. I am also exceptionally grateful to Nikolay Atanasov, who was a constant source of research inspiration and who helped me think through many difficult research problems throughout my Ph.D. This dissertation would not have been possible without either of their support.

Next, I thank the rest of my dissertation committee, Vijay Kumar and Hamed Hassani, who both provided valuable guidance and discussions of the robotics problems I expressed interest in. I also thank my closest collaborators, Dinesh Thakur, Vasileios Tzoumas, and Xiaoyi (Jeremy) Cai, for working so hard to bring our research ideas to fruition. I also thank my co-authors Heejin Jeong, Ke Sun, Alex Robey and Arman Adibi for valuable discussions and for giving me the opportunity to collaborate and participate in their research ideas.

I thank friends and labmates of the GRASP lab, Arbaaz Khan, Clark Zhang, Matt Cleaveland, James Svacha, Alex Zhu, Steven Chen, Ty Nguyen, Andreea Alexandru, Tasos Tsiamis, Dan Mox, Diego Caporale, and many others for interesting conversations and fun activities over the years.

I thank my undergraduate mentor Romel Gomez, and advisor P.S. Krishnaprasad for giving me my first research opportunities at the University of Maryland.

Finally I thank my family, Debbie, Steve, and Alec Schlotfeldt, my cousins Eva Klein and Sean Canfield, and most importantly Kate Tolstaya for being my partner and a constant source of inspiration throughout this journey.

ABSTRACT

DISTRIBUTED, SCALABLE AND RESILIENT INFORMATION ACQUISITION FOR MULTI-ROBOT TEAMS

Brent T. Schlotfeldt
George J. Pappas

Advances in robotic mobility and sensing technology have the potential to provide new capabilities in a wide variety of information acquisition problems including environmental monitoring, structure inspection, localization and mapping of unknown environments, and search and rescue, amongst many others. In particular, teams composed of multiple robots have shown great potential in solving these problems, though it is challenging to design efficient algorithms that are distributed and scale well, and even more complex in hazardous or challenging environments. The purpose of this dissertation is to provide novel algorithms to the capabilities of multi-robot teams to gather information which are distributed, scalable, and resilient.

The first part of the dissertation introduces the single-robot information acquisition problem, and focuses on algorithms that may be used for individual robots to plan their own trajectories. The methods presented here are search-based, meaning that an individual robot has a finite set of actions and is seeking to efficiently build a search tree over a known planning horizon. The first method presented details how to use the concept of algebraic redundancy and closeness to achieve a smooth trade-off of completeness in the exploration process, as an anytime planning algorithm. Next we show how a single robot can compute an admissible and consistent heuristic which guides the search towards the most informative regions of the state space, using the classic A* planning algorithm, drastically improving the search efficiency.

The next chapter of the dissertation focuses on how to build on the single robot plan-ning algorithms to create efficient algorithms for multi-robot teams, which operate in a distributed manner and scalable manner. The first method presented is coordinate descent,

otherwise known in the literature as sequential greedy assignment. This algorithm is implemented in a multi-robot target tracking hardware experiment. Next, we formulate an energy-aware multi-robot information acquisition problem, which allows for heterogeneity and captures trade-offs between information and energy expenditure. However, this results in a non-monotone objective function. Therefore we propose a new algorithm based on distributed local search, which achieves performance guarantees through a diminishing returns property known as submodularity.

The final chapter focuses on hazardous or failure prone environments that necessitate resilience to a fixed number of failures in the multi-robot team. We provide a definition of resilience, and formulate a resilient information acquisition problem. We then propose the first algorithm that solves this problem through an online application of robust trajectory planning, and provide theoretical guarantees on its performance. We then present three unique applications of the resilient multi-robot information acquisition framework, including target tracking, occupancy grid mapping, and persistent surveillance which demonstrate the efficacy of our approach.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A Motivation

Improvements in robotics technology often promise to increase the capabilities in automating of difficult and dangerous tasks. Examples are structure inspection, environmental monitoring, localization and mapping of unknown areas, surveillance, and search and rescue. There are instances such as the recent Fukushima disaster, where the environments are actually dangerous for humans to operate in, and therefore there is a great need to develop these capabilities. As the advances in mobility and sensing are increasingly cost effective, it becomes possible to solve these problems with not just a single highly equipped robot, but rather a team of lower cost and agile robots. These teams may be heterogeneous in their mobility (air vs ground vehicles), or their sensing modality (LIDAR or cameras).

Information gathering is a general formulation of the above mentioned problems which is designed to maximize a metric of uncertainty about a process that the robots are interested in measuring, subject to their unique sensing and mobility constraints. The problem is challenging for a number of reasons, but especially because common uncertainty measures are highly non-convex, and robot dynamics are often non-linear, which makes it difficult to plan trajectories efficiently for a single-robot, let alone a large team in a large-scale

environment, which demands non-myopic planning. Compounding these challenges are the need for algorithms that can operate in these hazardous environments where the robot team itself might be subject to failures; thus algorithms that are distributed and do not rely on a central node for communication are of critical importance. Applications of the problem are broad and include mobile robot target tracking [2, 1], Exploration and Mapping (including Simultaneous Localization and Mapping) [3, 4, 5], Monitoring of Hazardous Environments [2, 6], and Persistent Surveillance [7, 8, 9].

The focus of this dissertation is therefore to address the general information acquisition problem in several ways. The first is to provide efficient algorithms for non-myopic single-robot information gathering problems, challenging in their own right, which can be composed to form solutions to multi-robot problems. Given these building blocks, we consider using them to solve multi-robot information acquisition problems which allow for heterogeneous sensing and motion models, and we implement the algorithms in real-time hardware experiments to demonstrate their feasibility. The final chapter of the dissertation is focused on formulating and providing an efficient an accurate solution to the resilient information acquisition problem, which captures the potential for failures in the team in the problem formulation.

B Related Work

Currently, practical implementations of autonomous systems in the real world are passive in their information collection. New observations from on-board sensors are efficiently integrated into the robot's belief model of the world without actively being controlled. In contrast, humans and biological organisms actively seek out new information. This was first emphasized in the robotics literature in Ruzena Bajcsy's early work on Active Perception [10].

As advances in state estimation algorithms progressed, complex problems such as local-

ization and mapping were solved using probabilistic frameworks for modeling robot dynamics and sensor observations. This led to techniques such as the Extended Kalman Filter (EKF), or the Particle Filter algorithms [11]. In the EKF formulation of the localization and mapping problems, the target state to be estimated is formulated as a Gaussian random variable, which may include the robot pose as well as the location of a set of landmarks. Since the distribution remains Gaussian, there is a covariance matrix which may help quantify the current uncertainty and correlations amongst the landmarks and robot pose itself. Some of the earliest work on active localization [12, 13] proposes the use of entropy minimization over the belief distribution, to select meaningful actions. In the case of Gaussian distributions, the entropy is a direct function of the covariance matrix itself. In the case of non-Gaussian distributions such as those derived from a particle filter, a useful technique presented in [11] is to fit a Gaussian to the distribution, and then compute the entropy. The approaches presented at this point however are greedy in their nature, in that they do not consider optimizing over a long planning horizon, but instead look at one action at a time. This is because it is computationally challenging to reason over multiple timesteps, and possible evolutions of the belief state.

As the research matured in the active information acquisition literature, techniques such as [14, 15] proposed search based methods for solving information acquisition problems over a longer horizon, especially in problems where the target distribution remains Gaussian. These works showed the benefits of planning over longer horizons, which allow robots seeking information to avoid local minima that the greedy algorithms fall into.

Following the success of the early approaches to information acquisition with single robots, advances in miniaturization and sensing technologies made it practical to study multi-robot information acquisition, as a means to collect more and more observations and better solve information acquisition problems. The first works on multi-robot information acquisition used greedy planning and approximations of the mutual information objective function [16] to achieve a good level of performance in some tasks.

Some closely related fields that developed during this time are those of sensor selection, placement and scheduling [17, 18, 19, 20], which are problems concerned with selecting a set of sensors to observe a process. This differs from the information acquisition problem, because there is no internal state of the sensors that changes over time, namely the sensor observations depend directly on the sequence of control inputs given to the robots, which control their pose that they may generate observations from. Therefore the information acquisition problem is more challenging.

Two mathematical properties that are commonly leveraged in the literature on sensor selection and sensor placement are those of monotonicity, and submodularity. Monotonicity means that when additional sensors are added, the overall information objective cannot decrease [21]. In other words, more robots equals more information. Submodularity is a diminishing returns property [21] which says that, when additional robots are added, the marginal gain on information obtained is less than or equal to the gain that would be incurred if there were no other robots. In other words, there are diminishing returns to adding an increasingly high number of robots.

Most modern approaches for multi-robot information gathering can be grouped into methods that include search-based [2, 22], sampling-based [23, 24], and gradient-based planning [25]. The methods can also differ on problem parameters, such as the length of the planning horizon myopic (or one-step-ahead) planning [26] versus non-myopic (or long-horizon) planning [2]— and the type of target process (e.g., Gaussian [14], Gaussian mixture, [3, 27], occupancy grid map [5, 16], and non-parametric [28]).

Another recent area of research interest in multi-robot systems is the concept of resilience. Resilience may be defined as an ability for a system to recover from failure, and still complete a task. The specifics of how resilience is defined depend on the type of problem being solved, as well as the type of failures that might occur in the system. Recent works dealing with resilience in flocking are [29] and for resilience in target tracking [26].

Compared to the prior existing literature, this dissertation proposes new ideas for ef-

ficient single-robot planning which can compute plans faster, in real time, and for longer horizons. We then show an experimental hardware implementation of these ideas for a multi-robot problem and propose a distributed method that works on non-monotone energy-aware information acquisition problems, which previously has not been studied in a robotics and information acquisition context. Finally, we introduce the first problem for resilient information acquisition, and propose an efficient algorithm for its solution.

C Outline and Contributions

The overall objective of this dissertation is to develop the capability for large multi-robot systems to efficiently gather information over long planning horizons, in the presence of attacks and failures that compromise the system. To this end, we make the following contributions:

Chapter 2

This chapter explores two key ideas in the formulation of single-robot planning algorithms for information gathering, namely:

- Domination and Closeness Criteria (ε, δ)-redundancy
- Heuristics and Search-Based Planning

The first idea is to build search trees with nodes that can be pruned early if they are not showing a promising gain in information. In previous literature this notion has been explored, but we push the boundary further and develop an efficient anytime algorithm which iteratively refines the search procedure in order to guarantee a solution given finite run-time. This is critical for operation in a real-time planning system.

The second idea is to formulate the information gathering problem as a search-based planning problem, navigating towards a goal region, which is simply the planning horizon. The key observation is that, by developing a consistent and admissible heuristic which

captures the problem dynamics, the classic A* algorithm can be used to reach the final time horizon with the smallest uncertainty possible. A crucial element of this algorithm is to reduce the computation of the heuristic to derivation of upper bounds on the sensor information matrix, which we derive for position, range, bearing, and camera observation models.

We demonstrate the capabilities of these algorithms in target tracking and active mapping applications. Finally, we propose a method by which these two algorithms can be combined to yield an anytime planning algorithm that uses both the heuristic and domination criteria.

Chapter 3

This chapter describes how to compose the single robot planning algorithms from Chapter 2, in order to efficiently plan for multi-robot problems. The two algorithms presented here are:

- Coordinate Descent (Sequential Greedy Assignment)
- Distributed Local Search

Coordinate Descent, or Sequential Greedy Assignment as it is commonly referred to in the multi-robot literature, is a well known algorithm, and due to the mathematical properties of sub-modularity and monotonicity, it achieves a guarantee of 50% performance. We implement it here for real-time operation in a multi-robot target tracking hardware experiment, together with a distributed information filter. In combination with the single robot planners given in Chapter 2, we show that this algorithm is ready for real-time operation.

A downside of coordinate descent is that it relies on monotonicity for its performance guarantee. We introduce a new, but related problem in this chapter which considers energy-aware information acquisition for heterogeneous teams. Namely, problems where moving

an agent may cost more in energy than it gains in information. In these problems, the monotonicity property is lost, and therefore a new algorithm is needed. To combat this, we introduce the distributed local search algorithm, which provides a constant factor performance guarantee even for energy-aware problems, and still maintains a distributed nature.

In addition to the hardware experiments, we provide extensive simulations showing the capabilities of distributed local search, and it's improvement over coordinate descent for energy-aware problems.

Chapter 4

Finally, we put all the building blocks together from Chapters 2 and 3, and formulate a new problem that deals with a number of attacks or failures amongst the team. We call this problem Resilient Information Acquisition, which is a problem where the team of robots must be resilient to failures in the team, and still complete the information acquisition task. We show how the resilient problem may be solved by an online application of robust trajectory planning, where plans must be robust to a specified number of α unique failures in the team, where a failure means that a failed robot does not contribute any information to the rest of the team for the duration of it's attack. Computationally this adds an extra level of complexity, in dealing with a fixed number of attacks. We formulate the robust trajectory planning as a min-max optimization over worst case attacks on the system, where the property of submodularity again helps to provide constant factor performance guarantees in the presence of attacks or failures.

As part of our experimental validation, we investigate three key variables of problems in resilience. Namely, the number of attacks on the system, type of attack, and the effect of re-planning rate on the system. To look at these three variables, we formulate three information acquisition problems: multi-robot target tracking, occupancy grid mapping, and persistent surveillance. We also implement a simple hardware experiment showing qualitatively the effect of resilience on a two-robot, two-target tracking problem.

Chapter 2

Domination Criteria and Heuristics for Single Robot Information Acquisition

A Introduction

Significant advances in robot sensing and mobility have enabled the effective use of robot systems in environmental monitoring [30, 31], search and rescue [9], source seeking [32], and autonomous mapping [33, 34, 35], and other problems that require rapid and accurate information collection. The active information gathering problem is complex because it involves a combination of perception, estimation and inference, and the control of mobile sensors. There is significant work focusing on the estimation and scheduling aspect of the problem, and there exist near optimal methods for sensor placement and scheduling of static sensors [17, 18, 19, 20]. Despite these impressive results, there is far less work on controlling mobile sensing platforms to actively gather information. Of the works that do consider mobile sensors, many plan greedily or use short planning horizons [36, 37]. This chapter improves upon the existing single robot information acquisition algorithms, where the goal is to design nonmyopic control policies minimizing the uncertainty in the target state, conditioned on future measurements.

Related Work Approaches for mobile sensor information acquisition include [38, 3, 39, 40, 41, 42, 43, 44, 45]. [41, 38, 3], all use non-Gaussian representations for the target state, which requires approximating mutual information (MI). [44] uses a POMDP formulation, and approximates MI. In [45], a novel data-driven approach via imitation learning is developed. These works typically sacrifice the length of the planning horizon in order to use nonlinear sensor and target models during the planning process. In this work, we instead sacrifice some model accuracy via linearization in order to plan for longer horizons.

Another critical issue that is particularly relevant to robotics, is that time for deliberation is often limited. In the case of mobile robots, it is often necessary to apply control actions in fixed time intervals [46]. Although many of the above approaches can be tuned to run in shorter times, none of the prior works feature an anytime algorithm, which efficiently and progressively improves the quality of the trajectory until the solution is needed. The anytime algorithm presented in this chapter is closely inspired by ARA* [47], which is an efficient anytime extension to the A* path planning algorithm. The key ideas in ARA* are to progressively build the solution, and minimize redundant computations by saving previously computed results.

Contributions Our previous work [15] developed the Reduced Value Iteration (RVI) algorithm as a solution to the single sensor, nonmyopic planning problem, providing tunable parameters with suboptimality guarantees on the solution. In [48], the work was extended to multiple sensors, and a decentralized solution to the multi-robot planning problem (but not estimation) based on coordinate descent was shown with performance guarantees. In this paper:

(i) we develop an anytime version of RVI, capable of monotonically reducing the suboptimality gap of the solution while respecting real-time constraints,

(ii) We propose the first consistent heuristic function for information acquisition problems,

and define it in terms of upper bounds on the Sensor Information Matrix.

(iii) We derive the necessary upper bounds on the Sensor Information Matrix for the commonly used position, range, bearing, and camera sensing models.

(iv) We provide simulation results using an A* algorithm with our proposed heuristic, and show it can generate optimal solutions to the planning problem in less time than existing approaches.

B Problem Statement

Consider a team of n mobile robots, obeying the following motion models:

$$x_{i,t+1} = f_i(x_{i,t}, u_{i,t}), \quad i \in \{1, \dots n\} \tag{2.1}$$

where $x_{i,t} \in \mathcal{X}_i \cong \mathcal{R}^{n_{x_i}}$ is the n_{x_i}-dimensional state of robot i at time t, with metric $d_{\mathcal{X}}$, $u_{i,t} \in \mathcal{U}_i$ is the control action applied to robot i at time t, and the set $\mathcal{U}_i$ of possible control inputs for robot i is finite. The goal of the robots is to track the evolution of a target system with (unknown) state y_t and dynamics:

$$y_{t+1} = A_t y_t + w_t, \quad w_t \sim \mathcal{N}(0, W_t) \tag{2.2}$$

The operation of each sensor is described by the following sensor observation model:

$$z_{i,t} = H_{i,t}(x_{i,t}) y_{i,t} + v_{i,t}(x_{i,t}), \tag{2.3}$$

$$v_{i,t}(x_{i,t}) \sim \mathcal{N}(0, V_{i,t}(x_{i,t})), \tag{2.4}$$

where $z_{i,t} \in \mathcal{R}^{d_{z_i}}$ is the measurement obtained by robot i at time t, and $v_{i,t}(x_{i,t})$ is a sensor-state-dependent Gaussian noise, whose values are independent at any pair of times and

across the sensors. The observation model may be nonlinear in the robot state x_t but must be linear in the target state y_t. The latter requirement can be relaxed by linearizing a nonlinear sensor model around an estimate of the target state.

To simplify notation, we let $x_t := \begin{bmatrix} x_{1,t}^T ... x_{n,t}^T \end{bmatrix}^T$, $z_t := \begin{bmatrix} z_{1,t}^T ... z_{n,t}^T \end{bmatrix}^T$, $v_t := \begin{bmatrix} v_{1,t}(x_{1,t})^T ... v_{n,t}(x_{n,t})^T \end{bmatrix}^T$, and define $H(\cdot)$ appropriately such that $z_t = H_t(x_t)y_t + v_t(x_t)$. Let $\mathcal{X} := \mathcal{X}_1 \times ... \times \mathcal{X}_n$, and $\mathcal{U} := \mathcal{U}_1 \times ... \times \mathcal{U}_n$. The information available to the sensors at time t is denoted:

$$\mathcal{I}_0 = z_0 \quad \mathcal{I}_t := (z_{0:t}, u_{0:(t-1)}), t > 0 \tag{2.5}$$

The active information gathering problem is stated below.

Problem 1 (Active Information Acquisition) * Given initial sensor states $x_0 \in \mathcal{X}$, a prior distribution of the target states y_0, and a finite planning horizon T, choose a sequence of functions $\mu(\mathcal{I}_t) \in \mathcal{U}$, for $i = 1$,..., n, and $t = 1$,..., $T-1$, which optimizes the following:

$$\begin{aligned} \min_{\mu \in \mathcal{U}^T} \quad & J_T^{(n)}(\mu) := \sum_{t=1}^{T} \log\det(\Sigma_t) \\ \text{s.t.} \quad & x_{t+1} = f(x_t, \mu), \qquad t = 0, ..., T-1 \\ & \Sigma_{t+1} = \rho_{t+1}^e(\rho_t^p(\Sigma_t), x_{t+1}), \quad t = 0, ..., T-1 \end{aligned} \tag{2.6}$$

where $\rho_t^e(\Sigma, x)$ is the Kalman filter measurement update, and ρ_t^p is the Kalman filter prediction step, as follows:

$$\begin{aligned} \text{Predict:} \quad & \rho_t^p(\Sigma) := A_t \Sigma A_t^T + W_t \\ \text{Update:} \quad & \rho_t^e(\Sigma, x) := \Sigma - K_t(\Sigma, x) H_t(x) \Sigma \\ & K_t(\Sigma, x) := \Sigma H_t(x)^T (H_t(x) \Sigma H_t(x)^T + V_t(x))^{-1} \end{aligned}$$

It is known from [15] that the above is a deterministic optimal control problem, for

*We remark that problem 2 has an objective function proportional to minimizing the conditional differential entropy, $h(y_t|z_{1:t})$, where $h(Y) := \int p(y) log p(y) dy$ is the differential entropy of a continuous random variable.

which open loop control is optimal and the Kalman Filter is the optimal estimator.

C Anytime Planning

C.1 Preliminaries

To begin, we introduce the Forward Value Iteration algorithm (FVI) [14]. FVI solves problem 2 by constructing a search tree of the possible trajectories a robot can take starting from a tuple of state, covariance, cost (x_0, Σ_0, J_0) with $x_0 \in \mathcal{X}$, Σ_0 is the initial covariance matrix of the target state, and J_0 the initial cost. At each timestep t, all the states in the search tree are contained in the set S_t, starting with $S_0 = \{(x_0, \Sigma_0, J_0)\}$. Then, the set S_{t+1} is computed by evaluating the dynamics $x_{t+1} = f(x_t, u_t)$, Kalman Filter Riccati map $\Sigma_{t+1} = \rho^e_{t+1}(\rho^p_t(\Sigma_t, x_t, u_t))$, and cost $J_{t+1} = J_t + \log\det\Sigma_{t+1}$ on every pair $(x_t, \Sigma_t, J_t) \subset S_t$ and for each $u_t \in \mathcal{U}$, to generate new pairs $(x_{t+1}, \Sigma_{t+1}, J_{t+1})$. Notably this contains $\mathcal{O}(U^T)$ nodes in the final level of the tree, and is not feasible to compute in real-time for long horizons T.

In [15], the Reduced Value Iteration (RVI) algorithm was developed, and suboptimality bounds were derived, which rely on the following definitions:

Definition 1 (ε-Algebraic Redundancy [49]) Let $\varepsilon \geq 0$ and let $\{\Sigma_i\}_{i=1}^K \subset S^n_+$ be a finite set. A matrix $\Sigma \in S^n_+$ is ε-algebraically redundant with respect to $\{\Sigma_i\}$ if there exist nonnegative constants $\{\alpha_i\}_{i=1}^K$ such that:

$$\sum_{i=1}^{K} \alpha_i = 1 \quad \text{and} \quad \Sigma + \varepsilon I_n \succeq \sum_{i=1}^{K} \alpha_i \Sigma_i \tag{2.7}$$

Definition 2 (Trajectory δ-Crossing [15]) Trajectories $\pi^1, \pi^2 \in \mathcal{X}^T$ δ-cross at time $t \in [1, T]$ if $d_{\mathcal{X}}(\pi^1_t, \pi^2_t) \leq \delta$ for $\delta \geq 0$.

RVI uses the notions of ε-Algebraic Redundancy, and Trajectory δ-Crossing in order to decide when nodes in the search tree described above can be removed while maintaining

suboptimality. This has the effect of pruning the tree to keep a manageable number of trajectories for the robot to consider over long horizons. The pruning of S_t works as follows: A new set S'_t is constructed, which always contains at least the optimal node $(x_t^*, \Sigma_t^*, J_t^*) \subset S_t$. Then the algorithm checks all other nodes in S_t, and adds them to S'_t only if they do not δ-cross the current optimal solution, or if they do δ-cross, but the covariance is not ε-redundant with respect to the nodes already added to S'_t. Then S_t is assigned to S'_t. Intuitively this removes all nodes that come close together in space and have similar covariances.

C.2 Motivation for Anytime Planning

The RVI algorithm is effective for solving problem (2) over long planning horizons, and provides sub-optimality guarantees. However, the original RVI algorithm provides no guidance on how to choose (ε, δ) to ensure reliable runtime when the target state grows or the environment is complex. In problems like SLAM or target tracking, the target state grows larger as more landmarks or targets are discovered and the same ε will prune less nodes out of the search tree. This results in a varying runtime dependent on the size of the target state, y_t. Furthermore, any collaborative control strategy depends on receiving plans from other robots in predictable intervals. Any variance in computation time can compound with more robots, therefore it is critical for a real-time implementation to provide guarantees on the runtime of the planning algorithm.

To address these issues, we propose an anytime version of the RVI algorithm (ARVI), which is able to compute plans given a specified amount of time and removes the requirement of tuning the (ε, δ) parameters for the specific mission. This improves the operability of robots in practice, by ensuring the robots will always have a trajectory available. In addition, the algorithm allows for the most optimal plans to be selected in the multi-robot case, by splitting the time allocated to the set of robots planning jointly, based on the size of the group.

C.3 Anytime Reduced Value Iteration

The original RVI algorithm maintains only the set of states in the tree at each timestep, denoted S_t. At each timestep, the motion model is evaluated on each state in S_t to form the set S_{t+1}. Then, the set S_{t+1} is pruned according to the parameters (ε, δ) following criteria from definitions 1 and 2.

A key insight is the most costly operation in the algorithm besides the algebraic redundancy check is computing the Kalman Filter update step. Thus, an efficient anytime algorithm should be able to re-use these computations from prior iterations. We introduce the notation, $\mathcal{S} := \{S_0, S_1, ...S_T\}$ which is the full search tree containing all levels of the tree that have been computed, and is built progressively during the algorithm. In order to re-use computations from prior steps, it will be necessary to distinguish which states have been pruned from the search tree without discarding them in case they are needed later. ARVI achieves this through the use of two additional sets indicating which states are open for exploration, or are closed and do not need to be explored again. The sets are denoted by $\mathcal{O} := \{O_0, O_1, ...O_T\}$, and $\mathcal{C} := \{C_0, C_1, ...C_T\}$, respectively. The opened states are required to remain in the tree for guaranteeing an (ε, δ) sub-optimal solution, while the closed states are those which have already been expanded along each control action.

The ARVI algorithm consists of two parts, namely the Main procedure, and the ImprovePath call. Main is responsible for sweeping over the parameters and checking the remaining time on computation, while ImprovePath computes a trajectory from the parameters (ε, δ). Note that Main immediately runs an RVI with both (ε, δ) set to ∞. This has the effect of guaranteeing at least a greedy solution is computed. We denote the allocated planning time as T_{ARVI}, which should not be confused with the planning horizon T. The algorithm is presented here:

In summary, ARVI works by keeping persistent sets $\mathcal{S}$, $\mathcal{O}$, $\mathcal{C}$. The set $\mathcal{S}$ saves previously computed states, while $\mathcal{O}$ marks states that are 'open' and need to be explored. $\mathcal{C}$ indicates

Algorithm 1 Anytime Reduced Value Iteration (x_0, Σ_0, T_{ARVI})

$J_0 \leftarrow 0, S_0 \leftarrow \{(x_0, \Sigma_0, J_0)\}$. $S_t \leftarrow \emptyset$ for $t = 1, ...T$
$C_0 \leftarrow \emptyset$
$O_t \leftarrow S_t$ for $t = 0, ...T$
$\mathcal{S} \leftarrow \{S_0, ..S_T\}$, $\mathcal{O} \leftarrow \{O_0, ...O_T\}$, $\mathcal{C} \leftarrow \{C_0, ..C_T\}$
$\varepsilon \leftarrow \infty, \delta \leftarrow \infty$
$\{\mathcal{S}, \mathcal{O}, \mathcal{C}\} \leftarrow ImprovePath(\mathcal{S}, \mathcal{O}, \mathcal{C}, \varepsilon, \delta)$
Publish best solution from $\mathcal{O}[T]$
while Time Elapsed $\leq T_{ARVI}$ **do**
    Decrease (ε, δ)
    $\{\mathcal{S}, \mathcal{O}, \mathcal{C}\} \leftarrow ImprovePath(\mathcal{S}, \mathcal{O}, \mathcal{C}, \varepsilon, \delta)$
    Publish best solution J from $\mathcal{O}[T]$

Algorithm 2 ImprovePath ($\mathcal{S}$, $\mathcal{O}$, $\mathcal{C}$, ε, δ)

for $t = 1 : T$ **do**
    for all $(x, \Sigma, J) \in O_{t-1} \setminus C_{t-1}$ **do**
        $C_{t-1} \leftarrow C_{t-1} \cup \{x, \Sigma\}$
        for all $u \in \mathcal{U}$ **do**
            $x_t \leftarrow f(x, u)$, $\Sigma_t \leftarrow \rho_{x_t}(\Sigma)$
            $J_t \leftarrow J + \log\det(\Sigma_t)$
            $S_t \leftarrow S_t \cup \{(x_t, \Sigma, J_t))\}$
    Sort S_t in ascending order according to $\log\det(\cdot)$
    $O_t \leftarrow O_t \cup S_t[1]$
    for all $(x, \Sigma, J) \in S_t \setminus O_t$ **do**
        % Find all nodes in O_t, which δ-cross x:
        $Q \leftarrow \{\Sigma' | (x', \Sigma', J') \in O_t, d_{\mathcal{X}}(x, x') \leq \delta$
        if isempty(Q) **or not**(Σ is ε-alg. redundant wrt Q) **then**
            $O_t \leftarrow O_t \cup (x, \Sigma, J)$
return $\{\mathcal{S}, \mathcal{O}, \mathcal{C}\}$

the states that have already been expanded and are 'closed'. The following guarantees on the correctness and performance of ARVI hold.

Theorem 3 (ARVI) The following are satisfied by the ARVI algorithm:

1. When ImprovePath(ε, δ) returns with finite (ε, δ), the returned solution is guaranteed to be (ε, δ)-sub-optimal.
2. The cost $J_T^{(n)}$ of the returned solution decreases monotonically over time.
3. Given infinite time, $\mathcal{C} = \mathcal{O} \subseteq \mathcal{S}$, and $O[T]$ will contain the optimal solution to the planning problem.

Property (i) in Thm. 3 means that each time the ImprovePath subroutine is called in the ARVI algorithm, a bounded suboptimal solution is available. Property (ii) means that when more planning time is given to the algorithm, the solution improves monotonically. Lastly, (iii) guarantees that if infinite time is given, ARVI will return the optimal solution to equation (2). See the Appendix for the proof of Thm. 3.

D Heuristic for Search Based Information Acquisition

In section, we cast the Linear Gaussian Information Acquisition problem as a search-based planning problem, which allows us to use well-known algorithms such as Dijkstra and A* search. To effectively utilize the A* algorithm, which is a best-first search method that uses a heuristic to estimate the cost-to-go until the goal region, we propose a heuristic and prove its consistency and admissibility. Thus using this heuristic for A* search enables us to recover the optimal solution to the information acquisition problem. The derived heuristic depends on upper bounds of the sensor information matrix that comes from the Information Filter form of the Kalman filter. We derive the necessary upper bounds for a variety of sensor types, and demonstrate the overall effectiveness of our planning approach in comparison to existing search-based approaches that prune based on domination criteria.

E Problem Formulation

We stay consistent with the notation from the previous sections, but due to the notation of a heuristic function, we change the notation for the observation model to use $c(\cdot)$ The operation of each sensor is described by the following sensor observation model:

$$z_t = c(x_t, y_t) + v_t(x_t), \tag{2.8}$$

$$v_t(x_t) \sim \mathcal{N}(0, V_t(x_t)), \tag{2.9}$$

where $z_t \in \mathbb{R}^{d_{z_i}}$ is the measurement obtained by the robot at time t, and $v_t(x_t)$ is a sensor-state-dependent Gaussian noise, whose values are independent at any pair of times. Here we note that the sensor model is allowed to be a general nonlinear function of x_t and y_t.

We now re-state the problem from the previous section, so we may cast it as a search-based planning problem.

Problem 2 (Deterministic Information Acquisition) Given an initial sensor state $x_0 \in \mathcal{X}$, a Gaussian prior distribution of the target states (y_0, Σ_0), and a finite planning horizon T, choose a sequence of control inputs $u_{0:T-1} \in \mathcal{U}$ for $t = 1, \ldots, T$, which minimize the sum of stage costs c:

$$\begin{aligned} \min_{u_0:u_{T-1}} \; & J_T = \sum_{t=0}^{T-1} \log\det\Sigma_t \\ \text{s.t.} \quad & x_{t+1} = f(x_t, u_t), \qquad t = 0, ..., T-1 \\ & \Sigma_{t+1} = \rho^e_{x_{t+1}}(\rho^p_t(\Sigma_t)), \;\; t = 0, ..., T-1 \end{aligned} \tag{2.10}$$

where $\rho^e_x(\Sigma)$ is the Kalman filter measurement update, and ρ^p_t is the Kalman filter prediction

step, as follows:

$$\begin{aligned}
\text{Predict:} \quad & \rho_t^p(\Sigma) := A_t \Sigma A_t^T + W_t \\
\text{Update:} \quad & \rho_x^e(\Sigma) := (\Sigma^{-1} + M(x))^{-1} = F_x(\Sigma)\Sigma \\
& M(x) := C(x)^T V(x)^{-1} C(x) \\
& F_x(\Sigma) := I - K_x(\Sigma) C(x) \\
& K_x(\Sigma) := \Sigma C(x)^T R_x^{-1}(\Sigma) \\
& R_x(\Sigma) := C(x) \Sigma C(x)^T + V
\end{aligned}$$

where $M(\cdot) \in \mathbb{R}^{n_y \times n_y}$ is called the sensor information matrix.

In this work, we view the information acquisition optimal control problem as a planning problem. We state a standard definition of a planning problem for clarity:

Problem 3 (Planning Problem) Given a set of states S, an initial state $s_0 \in S$, a boolean function $G : S \to \{0,1\}$ which tells us whether the state is in the goal region, a function $A(s)$ which produces the valid control inputs in a given state s, a transition function $T(s,a) : S \times A(s) \to S$ for $s \in S$ and $a \in A(s)$, and a function $c(s,s') : S \times S \to \mathbb{R}$ returning the cost of being in state s and transitioning to state s', find a path $P = \{s_0, \ldots, s_n\}$:

$$\min_P \; J(P) = \sum_P c(s,s') \tag{2.11}$$

$$\text{s.t.} \;\; s_{i+1} = T(s_i, a_i) \tag{2.12}$$

$$G(s_n) = 1 \tag{2.13}$$

We now define the information acquisition problem as an instance of the planning problem. We define the planning state space to include both the spatial state and the information state of the information acquisition problem, in addition to the number of elapsed time steps since the initial time:

$$S := \{(x_t, \Sigma_t, t) \;\mid\; x_t \subseteq \mathcal{X}, \Sigma_t \succeq 0, t \geq 0\}$$

Then we can let the initial state s_0 be defined:

$$s_0 := (x_0, \Sigma_0, 0)$$

In the classic planning problem, the length of the path is unknown a priori and is determined as part of the search procedure. In this problem, we focus on a fixed horizon control problem, so our goal function can be defined:

$$G(s) := G(x_t, \Sigma_t, t) = \begin{cases} 1 & \text{if } t = T \\ 0 & \text{otherwise} \end{cases}$$

Next, we can define the available actions at any state to be the set of actions that give collision-free paths.

$$A(s) := A(x_t, \Sigma_t, t) = \{u \mid u \in \mathcal{U}\}$$

We also define the transition function that allows us to evaluate the next state given the current state and action:

$$\begin{aligned} T(s,a) :=& T(x_t, \Sigma_t, t, u_t) \\ =& [f(x_t, u_t), \rho^e_{f(x_t,u_t)}(\rho^p_t(\Sigma_t)), t+1] \end{aligned}$$

Finally, we define the state cost function to be the log determinant of the target covariance matrix:

$$c(s,a) := c(x_t, \Sigma_t, t, u_t) = \log\det\Sigma_{t+1}.$$

Deterministic shortest path problems such as Problem 3 can be solved using search or sampling-based methods. For example, A* is a best-first graph-search algorithm that works by expanding the most promising nodes in a search tree beginning at a root state. Let g to

be the cost incurred to reach the state (x_k, Σ_t, t) from the root:

$$g(x_t, \Sigma_t, t) = \sum_{k=0}^{t-1} \log\det\Sigma_k. \tag{2.14}$$

Next, we define a heuristic function $h(x_t, \Sigma_t, t)$, which serves as an estimate of the remaining cost-to-go along the path going through the state (x_t, Σ_t, t). An informative heuristic function massively improves exploration in a planning problem by delaying or ruling out regions of the graph which appear to be unpromising. If a planner has access to an optimal heuristic h^*, that is a heuristic which exactly equals the lowest possible cost-to-go from the desired state to the goal region, the planner will explore the exact optimal path and result in a linear time search. On the other hand, a trivial heuristic can be defined by $h(\cdot) = 0$. This would be considered an uninformative heuristic, since it provides no information about how promising or unpromising the current state is. Moving forward, our goal will be to design efficient and informative heuristic functions for the information gathering problem, since this will help remove unpromising regions from the search space.

A* is typically implemented with a priority queue, OPEN, where the order of state expansion is determined by the element in the priority queue with minimal p-value, which is the sum of the cost-to-come and the cost-to-go:

$$p(x_t, \Sigma_t, t) = \underbrace{g(x_t, \Sigma_t, t)}_{\text{cost-to-come}} + \underbrace{h(x_t, \Sigma_t, t)}_{\text{cost-to-go}}$$

Applying the algorithm above yields an optimal solution, provided the following two conditions hold on the heuristic function $h(\cdot)$:

- Admissibility: $h(s) \leq h^*(s) \;\; \forall s \in S$
- Consistency: $h(s) \leq c(s, s') + h(s') \;\; \forall s \in S$

Algorithm 3 A* Algorithm (x_0, Σ_0, T, $\mathcal{X}_g$)

g$(x_0, \Sigma_0, 0) = 0$; OPEN $= \emptyset$
Insert $(x_0, \Sigma_0, 0)$ into OPEN with $p(x_0, \Sigma_0, 0) =$ h$(x_0, \Sigma_0, 0)$
$(x_t, \Sigma_t, t) \leftarrow \text{argmin}_{(x_t, \Sigma_t, t) \in OPEN}\, p(x_t, \Sigma_t, t)$
while $(x_t, \Sigma_t, t) \notin \mathcal{X}_g$ **do**
    for For $u \in \mathcal{U}$ **do**
        $(x, \Sigma, k) = (f(x_t, u), \rho^e_{f(x_t,u)}(\rho^p_t(\Sigma_t)), t+1)$
        $g(x, \Sigma, k) = g(x) + \log\det\Sigma_t$
        $p(x, \Sigma, k) = g(x, \Sigma, k) + h(x, \Sigma, k)$
        Insert (x, Σ, k) into OPEN with $p(x, \Sigma, k)$
    $(x, \Sigma, t) \leftarrow \text{argmin}_{(x, \Sigma, t) \in OPEN}\, p(x, \Sigma, k)$

F Information Acquisition Heuristics

While the method detailed in Algorithm 3 is optimal, without an informative heuristic, the algorithm will default to Dijkstra's algorithm and will require the expansion of all nodes, causing the search complexity grow exponentially as $\mathcal{O}(|\mathcal{U}|^T)$. An informative heuristic dramatically speeds up the planning procedure, so we now focus our attention on deriving heuristics for the information acquisition problem.

F.1 Reduction to Bounding Sensor Information Matrix

A key observation we make in the problem structure is the relationship of the cost function to the Kalman filter covariance matrix $\Sigma_t \succeq 0$, and the sensor information matrix $M(x) = H(x)^T V(x)^{-1} H(x) \succeq 0$. Our objective is to reduce the computation of a heuristic to computation of an upper bound on the sensor information matrix over the reachable state space of the sensor.

Definition 4 (Reachable-Set) Given an initial sensor state x_0, the t-step reachable set can be

defined for $t > 0$ as:

$$\mathcal{R}^t(x_0) := \{x' \mid x' = f(x,u) \ \ \forall u \in \mathcal{U} \ \ \forall x \in R^{t-1}(x_0)\}$$

where $\mathcal{R}^0(x_0) := \{x_0\}$.

Lemma 5 Given a prior covariance matrix of a Gaussian distribution $\Sigma \succeq 0$, the Kalman filter prediction step $\rho_t^p(\Sigma) \succeq 0$. Moreover, given $0 \preceq M(x) \preceq \bar{M}^1(x)$, the following holds:

$$\rho_t^p(\Sigma_t) + M(x_t) \preceq \rho_t^p(\Sigma_t) + \bar{M}^1(x_t)$$
$$(\rho_t^p(\Sigma_t) + \bar{M}^1(x_t))^{-1} \preceq (\rho_t^p(\Sigma_t) + M(x_t))^{-1}$$

Theorem 6 (Heuristic Functions for Information Acquisition) Let the reachable set by the robot at state x in $t > 0$ timesteps be denoted by $\mathcal{R}^t(x)$. Suppose there exists a matrix $\bar{M}^t(\bar{x})$ such that $\bar{M}^t(\bar{x}) \succeq M(\bar{x}) \quad \forall \bar{x} \in \mathcal{R}^t(x)$. Then the following heuristic is consistent and admissible:

$$h(x_t, \Sigma_t, t) = \sum_{k=t}^{T} \log\det((\rho_k^p(\Sigma_k)^{-1} + \bar{M}^{k-t+1}(x_t))^{-1}),$$

where $\Sigma_{k+1} = (\Sigma_k^{-1} + \bar{M}^{k-t+1}(x_t))^{-1}$ and $(x_k, \Sigma_k, k) = (x_t, \Sigma_t, t)$

Proof 1 In [50], it is shown that consistency implies admissibility for heuristic functions. Therefore, we only need to show the proposed heuristic is consistent, which requires the following

$$h(x_t, \Sigma_t, t) \leq \log\det(\Sigma_{t+1}) + h(x_{t+1}, \Sigma_{t+1}, t+1)$$

We have:

$$
\begin{aligned}
&h(x_t, \Sigma_t, t) - h(x_{t+1}, \Sigma_{t+1}, t+1) \leq \log\det(\Sigma_{t+1}) \\
&= \sum_{k=t}^{T} \log\det((\rho_k^p(\Sigma_k)^{-1} + \bar{M}^{k-t+1}(x_t))^{-1}) \\
&- \sum_{k=t+1}^{T} \log\det((\rho_{k+1}^p(\Sigma_{k+1})^{-1} + M^{k-t+1}(x_t))^{-1}) \\
&= \log\det((\rho_t^p(\Sigma_t)^{-1} + \bar{M}^1(x_t))^{-1}) \\
&\underset{a}{\leq} \log\det((\rho_t^p(\Sigma_t)^{-1} + M(x_t))^{-1}) \\
&\underset{b}{=} \log\det(\Sigma_{t+1})
\end{aligned}
$$

Where (a) holds by monotonicity of $\log\det(\cdot)$, and by Lemma 1 since $M(x) \preceq \bar{M}^1(x)$. Then (b) holds by the Kalman Filter Riccati Map.

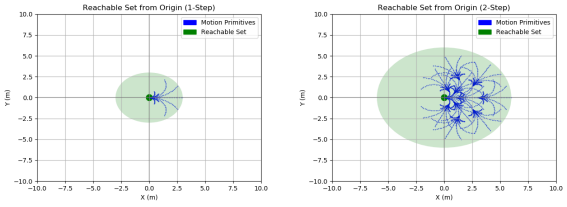
Corollary 7 (Optimality and Inflated Heuristic) The solution obtained by Algorithm 3 with the heuristic function proposed in Theorem 6 returns an optimal solution J_T^*. If the heuristic is scaled by a factor of ε, the returned solution J_ε has bounded sub-optimality such that $J_T^* \leq J_\varepsilon \leq \varepsilon J_T^*$.

Proof 2 The proof follows immediately from the consistency and admissibility properties proved in Theorem 6, and from the bounds on inflated heuristics obtained in [47].

Remark The result obtained in Theorem 6 holds for any monotone cost function of the covariance Σ_t, and is not restricted to $\log\det(\Sigma_t)$. For instance, other commonly used uncertainty measures such as the trace $tr(\Sigma_t)$ also lead to consistent heuristics.

F.2 Approximation of the Reachable Set

The implication of Theorem 1 is two-fold. First, it constructs a consistent, and admissible heuristic function that can be used to speed up the planning process. Second, it reduces



(a) Reachable set $\mathcal{R}^1(0,0)$. (b) Reachable set $\mathcal{R}^2(0,0)$.

Figure 2.1. Here we show the evolution of a finite set of motion primitives, and the approximate reachable set computation. Figure (a) shows one timestep, and Figure (b) shows two timesteps.

this heuristic computation to the problem of computing an upper bound on the sensor information matrix over a reachable set $\mathcal{R}^t$, i.e. $M(x) \preceq \bar{M}^t(x) \quad \forall x \in \mathcal{R}^t$.

A first step towards computing an upper bound on the sensor information matrix is to approximate the t-step reachable set $\mathcal{R}^t(x)$ from a given sensor state x. Because we consider finite action spaces, the true reachable set from a given configuration grows exponentially in the number of timesteps. Though there are more intricate methods to compute the reachable set of a dynamical system as detailed in [51], we resort to an over-approximation by a ball of finite radius $r_{\mathcal{U}}(x)$, with an increasing radius in the number of timesteps for its simplicity.

$$r_{\mathcal{U}}(x) = \max_{u \in \mathcal{U}} d_{\mathcal{X}}(x, f(x,u)) \tag{2.15}$$

Thus, an over-approximation $\bar{\mathcal{R}}^t(x) \supseteq \mathcal{R}^t(x)$ for the set reachable in t timesteps can be constructed as:

$$\bar{\mathcal{R}}^t(x) := \{\bar{x} \in \mathcal{X} \mid d_{\mathcal{X}}(\bar{x}, x) \leq t * r_{\mathcal{U}}(x)\} \tag{2.16}$$

Note that for this to be an over-approximation, some minor continuity assumptions must hold on the motion model. . See Figure 2.1 to visualize the construction of the approximate reachable set $\bar{\mathcal{R}}^t(x)$

Given a construction of a reachable set, we focus on the problem of computing a bounding sensor information matrix such that $M(\bar{x}) \preceq \bar{M}^t(\bar{x}) \ \forall \bar{x} \in \bar{\mathcal{R}}^t(x)$. Doing this requires examination of specific sensor models.

G Bounding the Sensor Information Matrix

In this section, we derive bounding sensor information matrices for several common sensors that are often used in essential robotics tasks such as localization and mapping problems, target tracking, and others. We begin by extending the reachable set concept to account for observability:

Definition 8 (Reachable-Observable Set) Given an initial sensor state x_0, a t-step reachable set $\mathcal{R}^t(x_0)$, and a function $O(x,y) : \mathcal{X} \times \mathbb{R}^{n_y} \to \{0,1\}$ indicating whether the target process y is observable from a sensor state x, we can define the reachable-observable set as:

$$\mathcal{O}^t(\mathcal{R}^t(x_0)) := \{y \mid O(x,y) = 1 \ \forall x \in \mathcal{R}^t(x_0)\} \tag{2.17}$$

The reachable-observable set includes all target states y which the sensor may observe in t-steps. Since we are only seeking upper bounds on the sensor information matrix, we can resort to an over-approximation of the reachable-observable set, similar to the method for approximating the reachable set itself. We assume here that a sensor has a maximum range for which it can observe a component of the target from $\bar{r}(y)$:

$$\begin{aligned} \bar{r}(y) = &\max_{x \in \mathcal{X}} d_{xy}(x,y) \\ &\text{s.t. } O(x,y) = 1 \end{aligned}$$

Now, to over-approximate the reachable-observable set, we can build on top of the reachable

set construction from Equation 2.16 as follows:

$$\bar{\mathcal{O}}^t(x) := \{y \mid O(x,y) = 1 \ \ \forall x \in \bar{\mathcal{R}}^t(x) \ \} \tag{2.18}$$

We now have all the machinery to derive bounds on the sensor information matrix for a variety of sensor types.

G.1 Unobservable Case

A simple but important case to begin with is when $y \notin \bar{\mathcal{O}}^t(x)$, that is a target is strictly non-observable in t-steps from a given sensor location. There is no possibility of obtaining a measurement in this case, the Kalman Filter resorts to a prediction only step. An alternate way to view this, is that the observation matrix $C(x)$ from sensor state x is simply zero. Since taking $C(x) = 0$ implies $M(x) = C(x)^T V^{-1}(x) C(x) = 0$, a trivial upper bound can be obtained as:

$$\bar{M}(x) = 0$$

when $y \notin \bar{\mathcal{O}}^t(x)$. The remaining sections now assume the target y is observable.

G.2 Position-Sensor

The position sensor reports the relative position of a point $y \in \mathbb{R}^3$ from a sensing location $x = (p, R)$ with position $p \in \mathbb{R}^3$, and orientation $R \in SO(3)$.

$$z = c(x,y) = R^T(y-p) + v, \quad v \sim \mathcal{N}(0, V(x)) \tag{2.19}$$

In the case where the point y is observable, the sensor observes a noisy estimate of the translated state $y - p$, where $C(x) = R^T$ is the orientation of the sensor in the world frame. In this case, we have that $M(x) = RV(x)^{-1}R^T$. We can bound $V(x)^{-1}$ above by it's maximum eigenvalue, i.e. $\lambda_{max} I_3 \succeq V(x)^{-1}$. This yields a bound of $M(x) = \lambda_{max} RR^T = \lambda_{max} I_3$, since

$RR^T = I_3$. Thus, we can derive the following bound on the sensor information matrix:

$$\bar{M}^t(\bar{x}) = \lambda_{max} I_3 \tag{2.20}$$

G.3 Range-Sensor

The range sensor reports the relative distance of a point $y \in \mathbb{R}^3$ from a sensing location $x = (p, R)$ with position $p \in \mathbb{R}^3$, and orientation $R \in SO(3)$.

$$z = c(x,y) = \|p - y\|_2 + v, \quad v \sim \mathcal{N}(0, \sigma_r^2) \tag{2.21}$$

The range sensor is a non-linear sensor model, which requires us to adapt our formulation. Previous work has approached this problem by linearizing and planning based on the Extended Kalman Filter (EKF) covariance, which we will consider here. The linearized observation model is:

$$\nabla_{y=\hat{y}} c(p,y) = C(x) = \frac{(\hat{y} - p)^T}{\|p - \hat{y}\|_2} \tag{2.22}$$

Here we note that the linearized $C(x)$ is a row vector, and the sensing noise covariance $V(x) = \sigma_r^2$ is a scalar due to the 1-dimensional measurement. Thus the sensor information matrix $M(x) = C(x)^T V^{-1}(x) C(x)$ is an outer product scaled by the inverse noise covariance σ_r^{-2}. This outer product has one eigenvalue $\lambda = \sigma_r^{-2} \|C(x)\|_2$, and the remaining eigenvalues are zero. Taking the norm of $C(x)$:

$$\|C(x)\| = \frac{1}{\|p - \hat{y}\|_2^2} (\hat{y} - p)^T (\hat{y} - p) \tag{2.23}$$

$$= \frac{\|\hat{y} - p\|_2^2}{\|\hat{y} - p\|_2^2} \tag{2.24}$$

$$= 1. \tag{2.25}$$

Thus we can upper bound the sensor information matrix as:

$$\bar{M}^t(\bar{x}) = \sigma_r^{-2} I_3 \tag{2.26}$$

G.4 Bearing Sensor

The bearing sensor reports the relative bearing of a point $y \in \mathbb{R}^2$ from a sensing location $x = (p, \theta)$ with position $p \in \mathbb{R}^2$, and orientation $\theta \in [-\pi, \pi]$.

$$z = c(x,y) = \tan^{-1}\left(\frac{y_2 - p_2}{y_1 - p_1}\right) - \theta + v, \;\; v \sim \mathcal{N}(0, \sigma_b^2) \tag{2.27}$$

As for the range-sensor, we note that this is a non-linear sensor and must be linearized to obtain covariance estimates using the EKF. The linearization about the point y is given:

$$\nabla_{y=\hat{y}} c(x,y) = C(x) = \frac{1}{\|p - \hat{y}\|_2^2}\begin{bmatrix} -(\hat{y}_2 - p_2) & (\hat{y}_1 - p_1) \end{bmatrix} \tag{2.28}$$

Because the sensor is one dimensional with inverse noise covariance given by σ_b^{-2}, it can be easily seen that the sensor information matrix $M(x) = \frac{\sigma_b^{-2}}{\|p-\hat{y}\|_2}$, by applying the result from above for the range sensor. The difficulty in computing this bound is the need to place a limit on how close the sensor and target can be to avoid the singularity in $M(x)$, owing to the fact that there is an extra division by $\|p - y\|_2$. To avoid this singularity, suppose we place a range limit of $\underline{r}$ to denote the minimum range a measurement can be taken, similar to the maximum sensing range requirement. Then the following bound holds for $y \in \bar{\mathcal{O}}^t(\bar{x})$:

$$\bar{M}^t(\bar{x}) = \frac{\sigma_b^{-2}}{\max\left\{\underline{r}, \min_{p \in \bar{\mathcal{R}}^t(\bar{x})} \|p - \hat{y}\|_2\right\}} I_2 \tag{2.29}$$

G.5 Camera Sensor

Lastly, we consider the camera sensor, which reports the pixel location $z \in \mathbb{N}^2$ in image coordinates of a point $y \in \mathbb{R}^3$ in 3-D space, given a camera pose $x = (p, R)$ with $p \in \mathbb{R}^3$ and $R \in SO(3)$, and an intrinsic camera matrix $K \in \mathbb{R}^{2\times 3}$.

$$z = c(x,y) = K\pi(R^T(y-p)+v), \;\; v \sim \mathcal{N}(0,V) \tag{2.30}$$

where $\pi(y) := \frac{1}{y^3}y$ is a projection function.

We can take the gradient of the observation model with respect to a linearization point: $\hat{y}$:

$$\nabla_{y=\hat{y}} c(x,y) = C(x) = K\pi'(R^T(y-p))R^T \tag{2.31}$$

$$\pi'(x) = \frac{1}{x_3^2}\begin{bmatrix} x_3 & 0 & -x_1 \\ 0 & x_3 & -x_2 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.32}$$

To simplify notation, we let $P = \pi'(R^T(y-p))$. The sensor matrix $M(x) = C(x)^T V(x)^{-1} C(x)$ can be written as:

$$M(x) = RP(x)^T K^T V(x)^{-1} KP(x) R^T \tag{2.33}$$

Let λ_{KV} be the maximum eigenvalue of the matrix $K^T V^{-1} K \succeq 0$. Then let λ_P be the maximum eigenvalue of the matrix $P^T P$.

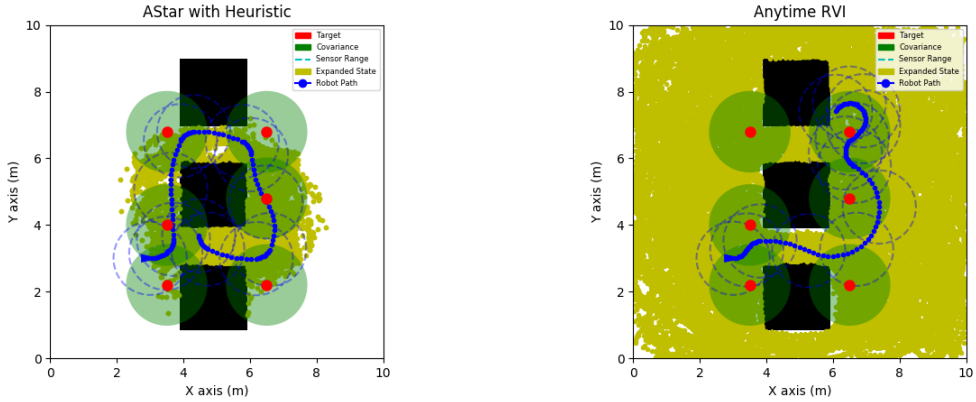
$$M \preceq \lambda_{KV}\lambda_P RR^T \tag{2.34}$$

$$\preceq \lambda_{KV}\lambda_P I_3 \tag{2.35}$$

The eigenvalues of $P^T P$ are $\lambda = \{1/x_3^2, \|x\|_2^2/x_3^4, 0\}$, so the maximum is always $(x_1^2+x_2^2+x_3^2)/x_3^4$. The final bound can be expressed as:

$$\bar{M}^t(\bar{x}) = \lambda_{KV} I_3 \max_{p \in \bar{\mathcal{R}}^t(\bar{x})} \frac{\|y-p\|_2^2}{(e_3 R^T (y-p))^4} \tag{2.36}$$

If the minimum depth approaching zero causes a singularity in the above expression, bounds can be introduced similar to the bearing sensor to ensure the information matrix remains finite.



(a) Expanded search tree with the proposed A* heuristic method.

(b) Expanded search tree for Anytime Reduced Value Iteration.

Figure 2.2. State expansions resulting from an A* algorithm using the proposed information acquisition heuristic (a) and Anytime Reduced Value Iteration algorithm [1] (b). Note that in (a), the state expansions are focused only in the areas where there are possible observations to be obtained about a landmark. In contrast, figure (b) shows state expansions covering the whole state space. Despite expanding more densely, the algorithm gets stuck in local minima observing only three of the six possible landmarks.

H Application to Active Mapping

H.1 Problem Setup

We apply the A* planning algorithm with the proposed information acquisition heuristic to an active mapping problem. We consider a scenario with six static landmarks with uncertain positions and a robot equipped with a range-only sensor aiming to minimize uncertainty

in the landmark position distribution (see Fig. 3.3). The robot follows differential-drive dynamics discretized with sampling period τ:

$$\begin{bmatrix} x^1_{t+1} \\ x^2_{t+1} \\ \theta_{t+1} \end{bmatrix} = \begin{bmatrix} x^1_t \\ x^2_t \\ \theta_t \end{bmatrix} + \begin{bmatrix} v\tau\,\mathrm{sinc}(\frac{\omega\tau}{2})\cos(\theta_t + \frac{\omega\tau}{2}) \\ v\tau\,\mathrm{sinc}(\frac{\omega\tau}{2})\sin(\theta_t + \frac{\omega\tau}{2}) \\ \tau\omega \end{bmatrix} \tag{2.37}$$

The action space is composed of discretized control inputs: $\{(v,\omega)|\ v \in \{1,3\}$ m/s, $\omega \in \{0,\pm 1,\pm 3\}$ rad/s$\}$.

The target state has dimension $n = 12$ as it is composed of the 2D landmark locations. The targets are assumed static:

$$y_{t+1} = y_t \tag{2.38}$$

Any of the previously described sensing models $c(x,y)$, corresponding to equations (2.19), (2.21), (2.27), or (2.30) could be used. Our example illustrates the behavior for a range sensor (2.21) with measurement noise standard deviation of $\sigma_r = 0.15$ m.

The joint measurement space consists of possible measurements for each landmark $m \in \{0,...M-1\}$, where M is the total number of landmarks being mapped. The linearized observation model for the joint target state can then be expressed as a block diagonal matrix,

$$C(x,y) = \begin{bmatrix} \nabla_y c(x,y_0) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \nabla_y c(x,y_{M-1}) \end{bmatrix} \tag{2.39}$$

Similarly, the bounding sensor information matrix $\bar{M}^t(\cdot)$ is the block diagonal matrix consisting of the blocks of each individual sensor information matrix for each target.

In the simulations, we let the sampling period be $\tau = 0.5$, which means the maximum

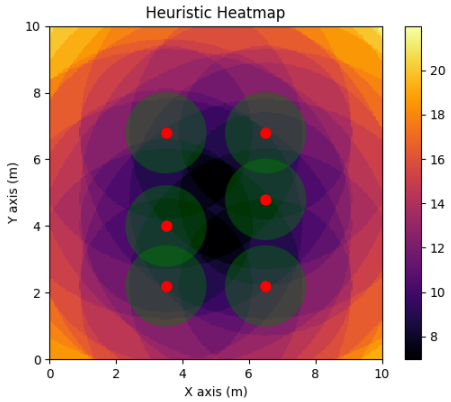


Figure 2.3. Heatmap of the heuristic function for the range-sensor, showing the estimated cost-to-go over the state space.

displacement from any action $r_{\mathcal{U}} = 3$ m/s $\times$ 0.5 (s) $= 1.5$ meters. Then the reachable set approximations are constructed from Equation 2.16. The maximum sensing range for the range sensor is 1 meter, with an omnidirectional field of view, which makes the reachable-observable set simple to evaluate given a target distribution. In cases with a limited field of view, the observable set can be over-approximated by an omnidirectional field of view for faster heuristic computation while still ensuring an upper bound on the sensor information matrix.

In Figure 2.3 we visualize the range sensing heuristic generated with a cost and heuristic based on the trace function: $J_T = \sum_{t=1}^{T} tr(\Sigma_t)$, for $T = 12$ timestep trajectories. The trace cost is used in the active mapping problem because it is less prone to leaving a target unobserved. Under determinant cost objectives, it is possible to find a path which localizes one target very well drastically reducing its minimum eigenvalue and thus the volume of the confidence ellipsoid, while other targets go unobserved.

We simulate the six target active mapping task, while varying the initial location of the sensing robot 100 times across the obstacle free space. To compare performance of the two algorithms, we consider the average cost of a node in the search tree generated by each algorithm. In Figure 2.4 we plot this distribution for the cost function $J = \sum_{t=1}^{T} trace(\Sigma_t)$.

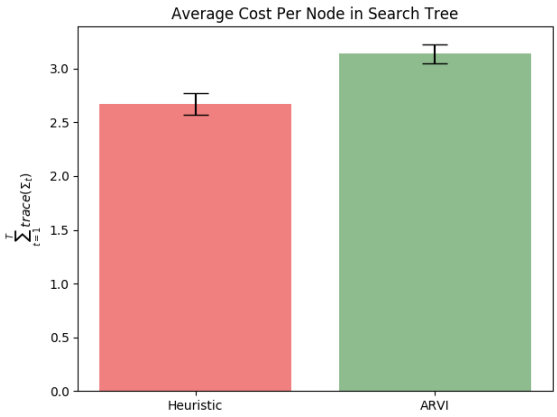


Figure 2.4. Graph showing the distribution of the average cost per path in the search tree for both A* (Mean=2.67) and ARVI (Mean=3.14).

A lower score on this metric indicates that an average node taken from a given search tree has a more optimal score, and thus the search tree has been constructed more effectively.

The heuristic cost map clearly demonstrates the idea that states whose t-step reachable sets may observe the target given the reachable set and observable set approximations, have a much lower heuristic value. The more targets that are observable from a given state, the smaller the remaining cost-to-go in a given state.

To visualize the resulting paths output from the A* algorithm with our proposed heuristic, we plot the planned paths for a single sensor tasked with localizing six landmarks. The prior uncertainty for each target is given as $\Sigma_0 = .25I_2$. The environment is a 10 meter by 10 meter region, containing three rectangular obstacles in the center. We visualize the sensing robot's planned trajectory and chosen observation points, along with the set of states which have been expanded during the construction of the search tree in Figure 3.3 (a).

H.2 Analysis

In Figure 3.3 (b) we compare our method with the Anytime Reduced Value Iteration algorithm, which is another algorithm for solving information acquisition problems of the form we proposed in [1]. That algorithm does not use a heuristic when constructing the

search tree, but instead attempts to iteratively construct the tree in a breadth first manner, pruning any nodes which satisfy a domination criteria, which removes nodes which early in the search are deemed unlikely to lead to an optimal path. The tree is built by expanding non-dominated node, in contrast to the A* algorithm which expands only the node with the most promise.

The results in Figure 3.3 depict a scenario where a sensing robot with the dynamics and sensing models described in the previous section needs to navigate around the obstacles to localize six landmarks. The A* method based on our proposed heuristic is able to find an optimal path which observes all six landmarks in less than 10 seconds, while the Anytime RVI algorithm fails to find a path that observes all the targets quickly enough, despite being given 60 seconds of execution time on a 2.4 GHz quad core CPU. This improved success in the active mapping task is due to the heuristic which intelligently constructs the search tree in areas where there are possible observations, while the ARVI algorithm spends too much search time in uninformative areas. The algorithms and heuristics are implemented with open source C++ and Python bindings at https://bitbucket.org/brentsc/infoplanner/src/iros_2019/.

The graph in Figure 2.4 shows that the average cost per node in the A* generated search tree is larger than the average cost of nodes in the ARVI generated search tree. This demonstrates quantitatively that the search tree is constructed more effectively, since on average it contains a set of nodes that lead to better trajectories for the sensing robot. We note here however, that due to the continuous motion model, the actual number of nodes can grow quite large in a given horizon. With an action space of size $\|\mathcal{U}\| = 10$, the search tree can grow in $T = 12$ steps to 10^{12}, or one trillion possible states, which is a limitation of not using the (ε, δ) pruning from the previous approach with continuous dynamics models.

I Conclusion: A Combined Approach

In this chapter, we showed a single robot formulation for the information acquisition problem, and develop efficient algorithms that solve it in two different ways. The first is a domination criteria, for which nodes can be pruned early in the construction of a search tree, to control the fullness of the tree. The second is a method of constructing a consistent and admissible heuristic, which guides the search to informative areas. For the first algorithm, we prove it's optimality given infinite run-time, and also show how it achieves a performance guarantee as a function of (ε, δ) parameters. For the second algorithm, we prove the heuristic is a function of the sensor information matrix upper bounds, and derive these upper bounds for position range, bearing, and camera sensors.

We conclude this chapter with presentation of a combined approach which we call ActivE Gaussian Information Search (AEGIS). AEGIS obtains the benefits of both approaches, namely it has controllable complexity in terms of (ε, δ), but also the search is guided using a heuristic. We present the algorithm, and a small illustration of it's implementation in a target tracking problem.

In Figure 2.5, we evaluate RVI, A*, and AEGIS on a simple target tracking problem with a range sensor, and system dynamics as in the previous section. All three algorithms ultimately achieve the same solution, but we can see the benefits of the combined approach. On the left, we see RVI, which explores a wide region even though it's unnecessary. The AStar solution looks good, but according to the (ε, δ) parameters there are some trajectories that do not need to be considered. In contrast, AEGIS efficiently explores towards the goal, while keeping the search space controlled with (ε, δ).

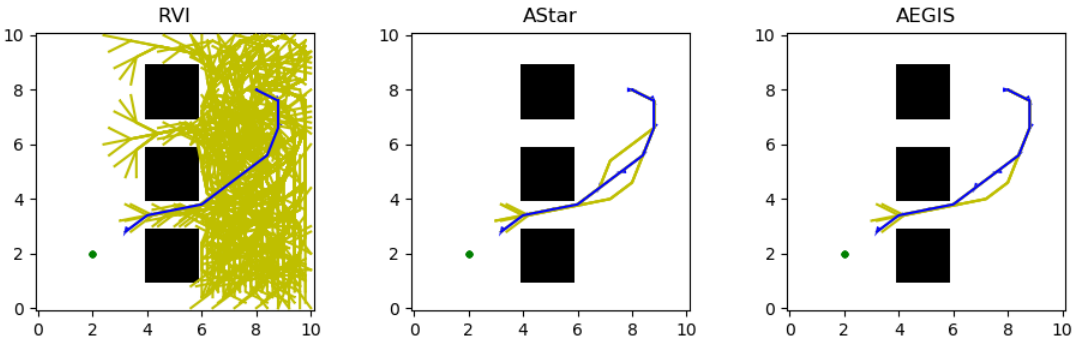


Figure 2.5. From left-right: RVI, AStar, and AEGIS. ($\varepsilon = 1, \delta = 0.1$)

Algorithm 4 ActivE Gaussian Information Search (AEGIS)

Require: initial state $(\mathbf{x}, \Sigma)$, initial cost J, initial time t, horizon T, domination parameters $\varepsilon, \delta \geq 0$, heuristic weight $w \geq 1$

function AEGIS($(\mathbf{x}, \Sigma, J)$, t, T, ε, δ, w)
  $g_t(\Sigma, \mathbf{x}) \leftarrow J$
  $\mathcal{O} \leftarrow (t, \mathbf{x}, \Sigma)$, $\mathcal{C} \leftarrow \emptyset$, $n_* \leftarrow nan$
  while $\mathcal{O} \neq \emptyset$ **and not** interrupted **do**
    $n \leftarrow \text{argmin}_{n' \in \mathcal{O}}\, g(n') + wh(n')$
    $\mathcal{O} \leftarrow \mathcal{O} \setminus \{n\}$
    if $n_* = nan$ **or** $g(n) + h(n) < g(n_*)$ **then**
      if ISDOMINATED(n, $\mathcal{O} \cup \mathcal{C}$, ε, δ) **then continue**
      $\mathcal{C} \leftarrow \mathcal{C} \cup \{n\}$
      for all $\mathbf{u} \in \mathcal{U}$ **do**
        $s, c \leftarrow$STEP(n, $\mathbf{u}$)
        if $g(n) + c + h(s) \geq g(n_*)$ **then continue**
        if $g(n) + c \geq g(s)$ **then continue**
        $g(s) \leftarrow g(n) + c$, parent$(s) \leftarrow (n, \mathbf{u})$
        if ISGOAL(s, T) **then** $n_* \leftarrow s$
        else if $s \in \mathcal{O}$ **then** HEAPIFY($\mathcal{O}$)
        else $\mathcal{C} \leftarrow \mathcal{C} \setminus \{s\}$, $\mathcal{O} \leftarrow \mathcal{O} \cup \{s\}$
  $\mathbf{u}_{t:T-1} \leftarrow$ control sequence leading from $(t, \mathbf{x}, \Sigma)$ to n_*
  if $\mathcal{O} = \emptyset$ **then** $error \leftarrow 0$
  else $error \leftarrow g(n_*) - \min_{n \in \mathcal{O}} g(n) + h(n)$
  return $\mathbf{u}_{t:T-1}$, $g(n_*)$, $error$

function ISDOMINATED($(t, \mathbf{x}, \Sigma)$, $\mathcal{S}$, ε, δ)
  for $(t', \mathbf{x}', \Sigma') \in \mathcal{S}$ **such that** $t' = t$ **and** $d_{\mathcal{X}}(\mathbf{x}, \mathbf{x}') \leq \delta$ **do**
    if $\Sigma + \varepsilon \succeq \Sigma'$ **and** $g_t(\Sigma, \mathbf{x}) + \varepsilon \geq g_t(\Sigma', \mathbf{x}')$ **then**
      return true
  return false

function STEP($(t, \mathbf{x}, \Sigma)$, $\mathbf{u}$)
  $\mathbf{x}' \leftarrow f(\mathbf{x}, \mathbf{u})$, $\Sigma' \leftarrow \Psi_{t+1}(\Phi_t(\Sigma, \mathbf{x}, \mathbf{u}), \mathbf{x}')$
  return $(t+1, \mathbf{x}', \Sigma')$, $\ell_{t+1}(\Sigma', \mathbf{x}')$

function ISGOAL($(t, \mathbf{x}, \Sigma)$, T) **return** $t = T$

Chapter 3

Multi-Robot Information Acquisition: Implementation and Energy-Aware

A Introduction

In this chapter, we turn our attention to the solution of multi-robot information acquisition problems. The use of multiple robots has been of much interest in a wide variety of applications such as autonomous mapping[34, 4, 52, 5], search and rescue[9, 53, 54], and environmental monitoring [55, 56, 57, 58]. These tasks require can be achieved more efficiently and accurately by larger robot teams, rather than relying on individual robots. Robot teams may take advantage of heterogeneous capabilities, require less storage and computation per robot, and may achieve better environment coverage in shorter time[59, 60, 61, 62]. Related work on multi-robot information gathering includes: [42, 4, 1, 24, 5, 63, 64, 65].

To alleviate the computational complexity which is exponential in the number of robots, various approximation methods have been developed to produce near-optimal solutions for a submodular and monotone objective (e.g.,, mutual information). A widely used technique is coordinate descent, or sequential allocation, where robots plan successively while incorporating the plans of previous robots[42, 4, 1, 24]. Ref. [42] showed that coordinate descent

extends the near-optimality of a single-robot planner to the multi-robot scenario. Later, [66] extended the result to dynamics targets, showing that coordinate descent solution attains at least 50% of the optimal performance regardless of the planning order. With a more sophisticated order, [63] used coordinate descent to decentralize the greedy method[67], by only committing the best single-robot trajectory to the team solution in each sequential round. Ref. [5] proposed a distributed algorithm, where robots first independently generate plans, and choose subsets of those plans in a sequential fashion to greedily optimize the objective for a fixed number of rounds.

To reduce the computational complexity of the multi-robot planning problem, it is necessary to decentralize the planning algorithm. [41] achieves this by computing MI only for pairs of sensors, decreasing the dimension of the required integration, while [40] assumes MI approximately decouples among groups of robots. Decentralization via coordinate descent is first proposed in [42].

Many approaches, for example [48] assume each robot has access to a centralized target state, which can be queried at any time. In this chapter, we assume each robot maintains its own estimate of the target state, and the robots perform a joint estimation step whenever they are in communication. The theory of our estimator is developed in [68], and is similar to the Kalman consensus filter [69, 70]. The filter is well-suited to the information gathering problem because it is able to work with Gaussian representations of the target, and it is mean-square consistent in the case of static targets, which is the case in some information acquisition problems such as environmental monitoring [30, 31].

Algorithms presented in Chapter 1 such as RVI and ARVI reduce the complexity in the planning horizon T, but the multi-sensor problem still scales exponentially with the number of robots n. We now formally introduce the Coordinate Descent approach to solving this problem, reducing the complexity from exponential to linear. Suppose that robot 1 plans its own trajectory with RVI, without considering the other robots. In other words, it solves

the single robot version of problem (2):

$$\mu^c_{1,0:(T-1} \in \underset{\hat{\mu} \in \mathcal{U}_1^T}{argmin} J_T^{(1)}(\hat{\mu}) \tag{3.1}$$

The robot then communicates its chosen plan, which may involve passing the control sequence and any of the models evaluated along the trajectory onwards to robot 2. Then robot 2 solves a two-sensor active information gathering problem, but assumes the fixed policy for robot 1:

$$\mu^c_{2,0:(T-1} \in \underset{\hat{\mu} \in \mathcal{U}_2^T}{argmin} J_T^{(2)}(\mu^c_{1,0:(T-1)}, \hat{\mu}) \tag{3.2}$$

The algorithm continues in this fashion, so that robot i needs the control sequences and the models evaluated along the trajectories of sensors 1 to $i-1$, solving an i-sensor version of equation (2). This algorithm reduces the planning complexity from exponential to linear, i.e. from $\mathcal{O}(|\mathcal{U}_1 \times ... \times \mathcal{U}_n|^T)$ to $\mathcal{O}(|\sum_{i=1}^n |\mathcal{U}_i|^T)$. It is proven in [15] that the solution J_T obtained from coordinate descent achieves at least 50% of the optimal value, i.e. $J_T \geq J_T^* \geq 2J_T$ when the objective function is negative mutual information with a similar result for conditional entropy.

A.1 Distributed Estimation

Although the coordinate descent scheme is decentralized in control, it assumes the robots have access to a centralized information source while planning. In many real-world problems that face challenges with regard to communication, this is not a realistic solution. Therefore, it is crucial to incorporate a distributed estimation algorithm into the multi-sensor active information acquisition problem. The estimator should naturally work with Gaussian target state representations, and have strong performance guarantees for static targets. Static targets are common in information acquisition problems like environmental monitoring and SLAM, and often these filters perform well in practice for dynamic targets.

We adopt a type of distributed Kalman filter first proposed in [68], and has the following update rule:

$$\begin{aligned} p_{i,t+1}(y) &= \zeta_{i,t} p_i(z_{i,t+1}|y) \prod_{j \in \mathcal{N}_i \cup i} (p_{j,t}(y))^{K_{i,j}} \\ \hat{y}_i(t) &\in \operatorname*{argmax}_{y \in \mathcal{Y}} p_{i,t}(y) \end{aligned} \tag{3.3}$$

where $\zeta_{i,t}$ is a normalization constant to ensure $p_{i,t+1}$ is a proper pdf, and $\kappa_{i,j}$ are weights such that $\sum_{j \in \mathcal{N}_i \cup \{i\}} \kappa_{i,j} = 1$. The filter update rule is the same as the standard Bayes rule except that each sensor i uses a geometric average of its neighbors' priors.

When specializing the estimator in (3.3) to the linear Gaussian measurement model we have used thus far, it is necessary to operate in the information space, which is equivalent to the covariance space [11], but offers some unique benefits. One advantage of filtering in the information space is that information is additive, and it is possible to have information vectors and matrices equal to zero. Most importantly however, the information space is sparse, while the covariance matrix is always dense. This results in significant computational savings, particularly when the dimension of the target state being estimated is large, as is the case when there are many targets. We introduce the quantities, ω and Ω, called information vector and information matrix respectively. Define $\Omega = \Sigma^{-1}$, and $\omega = \Omega y$. We have the following update-prediction rules:

$$\begin{aligned} \text{Update Step:} \quad \omega_{i,t+1} &= \sum_{j \in \mathcal{N}_i \cup \{i\}} \kappa_{i,j} \omega_{j,t} + H_i^T V_i^{-1} z_i(t) \\ \Omega_{i,t+1} &= \sum_{j \in \mathcal{N}_i \cup \{i\}} \kappa_{i,j} \Omega_{j,t} + H_i^T V_i^{-1} H_i \\ \hat{y}_i(t) &:= \Omega_{i,t}^{-1} \omega_{i,t} \\ \text{Predict Step:} \quad \Omega_{i,t+1} &= (A \Omega_{i,t+1}^{-1} A^T + W)^{-1} \\ \omega_{i,t+1} &= \Omega_{i,t+1} A \hat{y}_{i,t+1} \end{aligned} \tag{3.4}$$

where $H_i := H_i(x_i)$, and $V_i := V_i(x_i)$. Then, we define a communication network, indicating the connections of the sensors G = (V,E). The graph is fully connected if each sensor can communicate with each other. In the case of static targets, the following theorem holds:

Theorem 9 ([68]) Suppose that the communication graph G is connected and the matrix $\left[H_1^T ... H_n^T\right]^T$ has rank d_y. Then, the estimates in (16) of all sensors converge in mean square to y, i.e. $lim_{t\to\infty}\mathbb{E}[||\hat{y}_i(t) - y||_2^2] = 0$ for all i.

While the assumptions of Thm. 9 hold when all sensors are in communication, in practice it is likely that the sensor network is intermittently disconnected. Despite this, there are results [71] that suggest that filters of this type converge even under the weaker condition of an infinitely often connected network. This fact motivates the simulations in Sec. B which investigate the effects of limited communication ranges on the estimation performance.

B Hardware Implementation of ARVI and Coordinate Descent

B.1 Target Tracking Model

We now adapt the general models presented thus far for the target tracking application. Each sensor uses differential drive dynamics, discretized with a sampling period τ:

$$\begin{pmatrix} x_{t+1}^1 \\ x_{t+1}^2 \\ \theta_{t+1} \end{pmatrix} = \begin{pmatrix} x_t^1 \\ x_t^2 \\ \theta_t \end{pmatrix} + \begin{pmatrix} \nu\,\mathrm{sinc}(\frac{\omega\tau}{2})\cos(\theta_t + \frac{\omega\tau}{2}) \\ \nu\,\mathrm{sinc}(\frac{\omega\tau}{2})\sin(\theta_t + \frac{\omega\tau}{2}) \\ \tau\omega \end{pmatrix} \tag{3.5}$$

The control commands use motion primitives $\{(\nu,\omega)\} \mid \nu \in \{1,3\}$ m/s, $\omega \in \{0,\pm 1,\pm 3\}$ rad/s.

The targets move with discretized double integrator dynamics, and Gaussian noise where $y \in \mathbf{R}^n$ is the target state. For m targets, the size of the state is $n = 4m$, containing the planar coordinates and velocities denoted (y^1, y^2, $\dot{y^1}$, $\dot{y^2}$), of all targets. The model used is

the following:

$$y_{t+1} = A \begin{bmatrix} I_2 & \tau I_2 \\ 0 & I_2 \end{bmatrix} y_t + w_t, \quad w_t \sim \mathcal{N}\left(0, q \begin{bmatrix} \tau^3/3 I_2 & \tau^2/2 I_2 \\ \tau^2/2 I_2 & \tau I_2 \end{bmatrix}\right)$$

Note that a limitation of this approach is the requirement of having a model for the target a priori. A more robust approach may be to estimate the target model with reinforcement learning techniques, such as in [72, 73].

The sensor observation model, consists of range and bearing for each target $m \in \{0, ...M-1\}$, where M is the total number of targets in the environment.

$$\begin{aligned} z_{t,m} &= h(x_t, y_{t,m}) + v_t, \quad v_t \sim \mathcal{N}\left(0, V(x_t, y_{t,m})\right) \\ h(x,y) &= \begin{bmatrix} r(x,y) \\ \alpha(x,y) \end{bmatrix} := \begin{bmatrix} \sqrt{(y^1 - x^1)^2 + (y^2 - x^2)^2} \\ \tan^{-1}((y^2 - x^2)(y^1 - x^1)) - \theta \end{bmatrix} \end{aligned} \tag{3.6}$$

It should be noted again here that this model needs to be linearized, which can be achieved by taking the gradient with respect to a predicted target trajectory y, such that $y \neq x$.

$$\nabla_y h(x,y) = \frac{1}{r(x,y)} \begin{bmatrix} (y^1 - x^1) & (y^2 - x^2) & 0_{1x2} \\ -\sin(\theta + \alpha(x,y)) & \cos(\theta + \alpha(x,y)) & 0_{1x2} \end{bmatrix} \tag{3.7}$$

The observation model for the joint target state can then be expressed as a block diagonal matrix,

$$H(x,y) = \begin{bmatrix} \nabla_y h(x, y_0) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \nabla_y h(x, y_{M-1}) \end{bmatrix} \tag{3.8}$$

The sensor noise covariance grows linearly in range and bearing up to σ_r^2, and σ_b^2, where

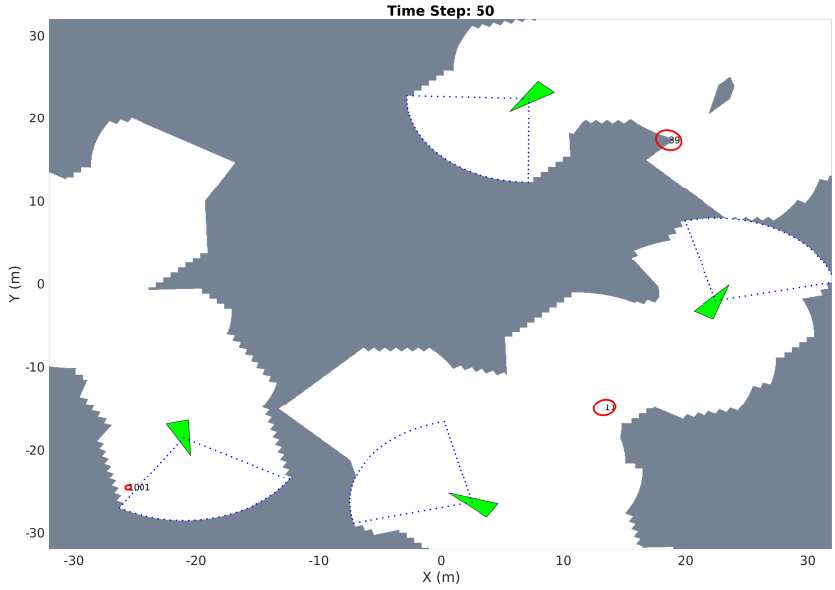


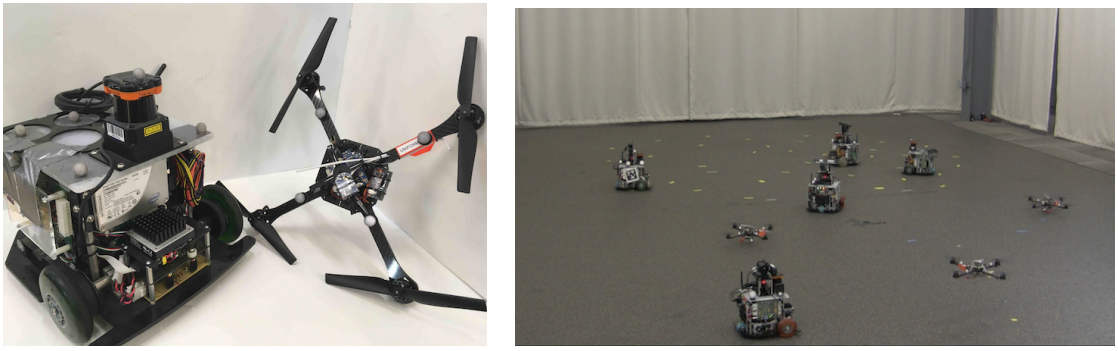
Figure 3.1. Simulation environment showing four robots (green) and three estimated target positions with covariance ellipses (red). Explored and unexplored regions are shown in white and gray color, respectively.

σ_r, σ_b are the standard deviation of the range and bearing noise. The model here also includes a limited range and field of view [74], denoted by parameters r_{sense}, and ϕ. If at any time, $r_m(x,y) \geq r_{sense}$, or $\alpha_m(x,y) \geq \frac{\phi}{2}$, the filter performs only a prediction step for target m, since no measurement is attainable when the target is outside the range or field-of-view limits.

Lastly, because the robots know neither the total number of targets, or have any prior information about the targets, we follow an approach from [75] and introduce an exploration strategy. This is achieved through the use of "exploration" landmarks with locations $l := [l_1^T, ...l_{N_l}^T]^T$. The landmarks are given a Gaussian prior with mean $\bar{l} := l$, and block diagonal covariance Σ^l. The landmarks are placed at the map frontiers which are maintained by a visibility occupancy grid.

B.2 Target Tracking Algorithm

Our multi-robot target-tracking algorithm is summarized:



(a) (b)

Figure 3.2. (a) A Scarab ground robot and a Hummingbird quadrotor. (b) The experimental setup with 3 UAVs and 5 Scarabs.

Algorithm 5 Anytime Multi-Robot Target Tracking

Input: $T_{max}, x_0, \hat{\omega}_0, \hat{\Omega}_0, f, \mathcal{U}, H, V, A, W, T, T_{ARVI}, n$
while $t = 1 : T_{max}$ **do**
    Send $\omega_{i,t}$ and $\Omega_{i,t}$ to neighboring robots.
    Receive measurements $z_{i,t}$ and perform distributed update step with any neighbor information received.
    Remove discovered exploration landmarks l and add new ones at map frontiers.
    Plan T-step trajectories by solving (2) with ARVI (Alg. 1) and coordinate descent.
    Apply first n controls to move each robot via f.

Alg. 5 was implemented in C++, and applied in both simulations and robot experiments. An open-source implementation* of the simulator and the algorithm is published online.

B.3 Simulation Results

In the simulation environment, the algorithm is evaluated in a virtual square region, with height and width of 64 meters (Fig. 4.2). Targets are initialized in random positions with zero velocity. The goal of the simulations was to evaluate the effects of communication radius, planning time, and robustness of the system to sensor noise. We test with a variable number of robots, targets, and parameters in each simulation, detailed in the captions. The performance metrics are the average root mean square error (RMSE) of target position, average entropy, and time to discover all targets.

As one may expect, increasing the communication radius (Fig. 3.3 (a-c)) results in better performance. We note that even a modest communication range of 5 meters provides substantial benefit, particularly in the rate of discovering targets. Without communication, a single robot is unable to find all the targets even over the course of 1000 timesteps. The benefits of increased planning time (Fig. 3.3 (d-f)) are noted by the reduction in position RMSE as well as average entropy. Interestingly, more planning time does not necessarily lead to discovering the targets faster. This is due to the planner's emphasis on keeping a low uncertainty on the targets currently known rather than discovering new targets. If fast exploration is desired, more uncertainty can be added to the exploration landmarks to encourage the robots to explore more rapidly at the expense of tracking the already known targets. Another observation is that the entropy and tracking error are similar in planning time at first, but start to differ as the simulation evolves. This is because the targets are initialized with zero velocity, and eventually start to move faster and require a better plan to track them. Finally, the performance in tracking and average entropy degrade with increasing sensor noise as might be expected. In summary, we see that target discovery is

*`https://bitbucket.org/brentsc/infoplanner`

tied to the availability of communication, while tracking performance depends on sensor reliability and the ability to plan non-myopic trajectories efficiently.

B.4 Hardware Experiments

We evaluate the real-time performance of the ARVI and Coordinate Descent algorithm on three collaborating UAVs, whose task is to explore a lab environment and find and track the locations of five ground robots (Fig. 3.2). The robot trajectories are planned using ARVI offboard on a laptop with an Intel Core i7 CPU. The Vicon Motion capture system is used for localization of both the ground robots and the UAVs. The ground robots are Scarab differential drive robots with top speed of 1.4 m/s [76]. The quadrotors were allowed motion primitives $\{(v, \omega) \mid v = 1 \text{ m/s}, \omega \in \{0, \pm 1, \pm 3\} \text{ rad/s}\}$. The target motion model in the hardware experiments was a random walk (e.g., $y_{t+1} = y_t + w_t$), instead of the double integrator. This is more suitable because of the sudden changes in velocity that the Scarab robots experience due to their collision avoidance algorithm. Sensor measurements are generated via the Vicon pose estimates, with a 360 degree field of view which imitates a downward facing camera. The measurements provide the relative pose of each ground robot within a sensing radius plus Gaussian noise, to each UAV. The testing area was a 4 by 8 meter region where the ground robots can move independently and freely to randomly generated waypoints. The ground robots utilize [77] to reactively replan safe paths around any static obstacles and other ground robots, while the quadrotors fly at different heights to avoid collisions. Ten separate five-minute trials were conducted, in which the UAVs tracked the ground robots and explored the environment. The performance of the algorithm can be seen in figure 3.4.

The primary purpose in running these experiments on real hardware was to ensure that the algorithm is able to compute trajectories reliably and run the decentralized algorithms with multiple robots, while also achieving a low tracking error. The original RVI algorithm paid no attention to execution time and without proper tuning it would take too long for

the robots to compute plans to fly in real-time. Even if RVI was tuned to always execute in a short time, it would not necessarily find the best trajectory in the allotted planning time, since it might terminate too early. To this end, the experiments were successful and show that ARVI achieves real-time search based planning for information gathering. In order to deploy the system in the real-world, there is still a need for incorporating collision avoidance in the planning process. More insight could also be gained by testing with a camera observation model, and planning on-board the UAVs.

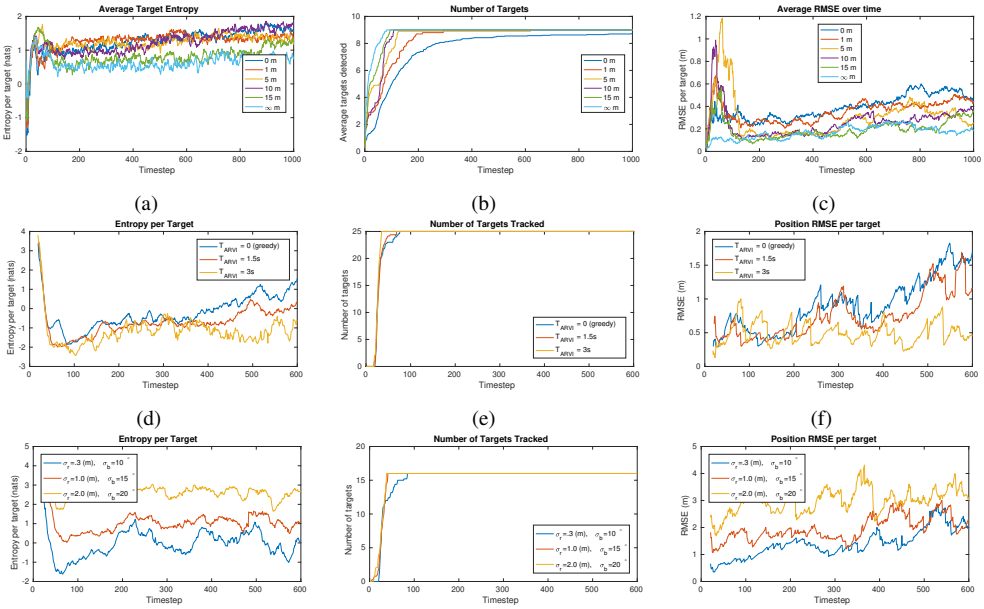
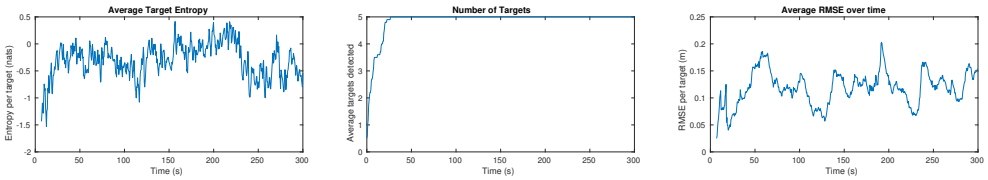


Figure 3.3. By column, the plots show the average entropy per target, number of targets discovered, and average position (RMSE) per target respectively. The top row uses 4 robots and 9 targets, with T_{max} = 1000, T_{ARVI} = 0.5s, σ_r = .15 m, σ_b=5°, and sweeps over communication range. The middle row uses 10 robots and 25 targets, T_{max} = 600, σ_r = .3 m, σ_b=10°, communication range of 30 meters, and sweeps over increasing T_{ARVI}. The last row uses 8 robots and 16 targets, with communication range of 30 meters, T_{ARVI} 0.5s, and sweeps over σ_r, σ_b. In all cases, ten Monte-Carlo trials are averaged for each parameter, with τ = 0.5s, $T = 12$, $n = 6$ controls before re-planning, r_{sense} = 10 meters, 94° field-of-view, and q = .001 fixed for every trial.



(a) (b) (c)

Figure 3.4. The plots show the average entropy per target, number of targets discovered, and average root mean square error (RMSE) per target respectively, all averaged over ten trials of five minutes. In the experiments, a planning horizon $T = 12$, and sampling period of $\tau = 0.5$ is used, along with $T_{ARVI} = 0.5$ split evenly. The communication range between UAVs is fixed at 3 meters. The sensing parameters used here correspond to a downward facing camera model, with $r_{sense} = 1$ meter, 360° field-of-view, with range and bearing standard-deviation of 0.3 m, and 5° respectively.

C Energy-Aware Information Acquisition

Practical implementations of multi-robot systems often need to consider various measures for energy expenditure, such as control effort, fuel cost, or distance travelled. A common approach is to impose fixed budgets, which preserves submodularity and monotonicity of the objective, so that existing algorithms may still be used [42, **?**, 63].

In this section, we are motivated by scenarios where robots, with potentially different sensing and control capabilities, have sufficient energy to spend but seek a desired trade-off between information gain and energy cost. Specifically, we formulate an energy-aware active information acquisition problem, where the goal is to plan trajectories for a team of heterogeneous robots to maximize a weighted sum of information gain and energy cost. One key observation with this approach is that adding the energy cost breaks the monotonicity of the objective, violating an assumption held by existing approximation algorithms. Therefore, we propose a new distributed planning algorithm based on local search [78] (see Fig. 3.5) to achieve a worst-case performance guarantee for the non-monotone objective. We further develop techniques to reduce its computation and communication requirements compared to its naive distributed implementation, in order to scale to larger robot teams without affecting its theoretical guarantees.

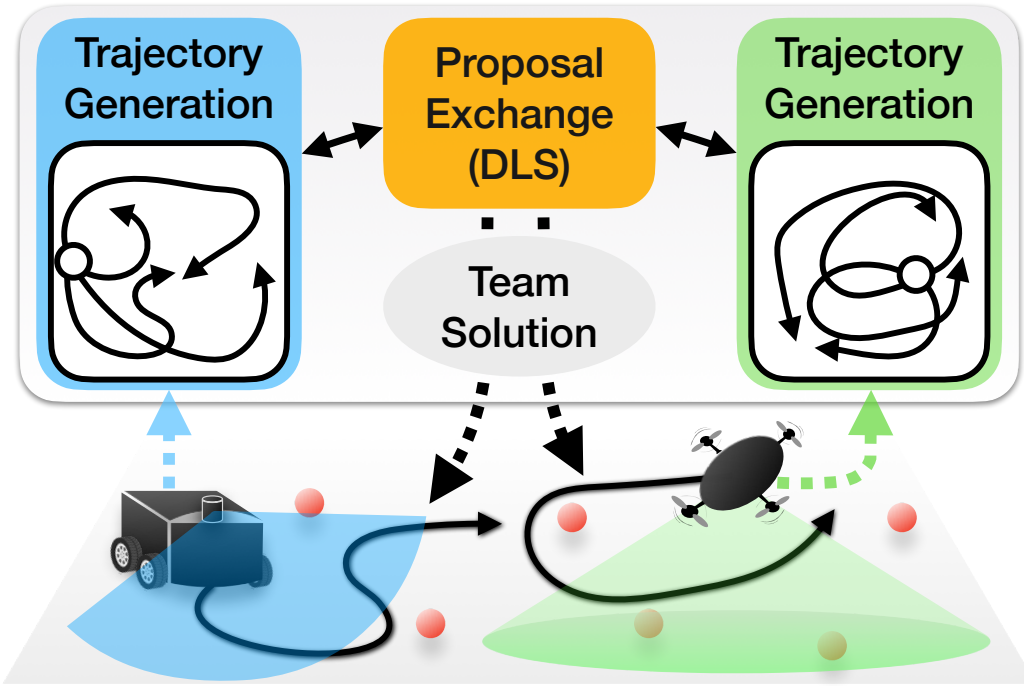


Figure 3.5. Overview of the proposed distributed planning approach for non-monotone information gathering (see Sec. C.3). Robots generate individual candidate trajectories and jointly build a team plan via distributed local search (DLS), by repeatedly proposing modifications to the collective trajectories.

Contributions. The main limitation of the prior works is the assumption of monotonicity of the objective function. Problems without monotonicity, such as the energy-aware problem we propose, cannot be solved by the above methods while retaining their near-optimality properties. In contrast, our proposed algorithm provides a theoretical performance guarantee even for non-monotone objectives. In this section:

- We propose a distributed algorithm based on local search where robots collaboratively build a team plan by proposing modifications to the collective trajectories;
- We reduce its computation and communication requirements without affecting the performance guarantee;
- We demonstrate that the proposed algorithm outperforms a state-of-the-art algorithm for multi-robot target tracking in coordinating a team of heterogeneous robots, while trading

off sensing performance and energy expenditure.

C.1 Preliminaries

We briefly review some useful definitions. Let $g : 2^{\mathcal{M}} \to \mathbb{R}$ be a set function defined on the ground set $\mathcal{M}$ consisting of finite elements. Let $g(a|S) := g(S \cup \{a\}) - g(S)$ be the discrete derivative, or the marginal gain, of g at S with respect to a.

Definition 10 (Submodularity) The function g is submodular if for any $S_1 \subseteq S_2 \subseteq \mathcal{M}$ and $a \in \mathcal{M} \backslash S_2$, $g(a|S_1) \geq g(a|S_2)$.

Definition 11 (Monotonicity) The function g is monotone if for any $S_1 \subseteq S_2 \subseteq \mathcal{M}$, $g(S_1) \leq g(S_2)$.

C.2 Problem Formulation

Consider robots indexed by $i \in \mathcal{R} := \{1, \ldots, n\}$ with states $x_{i,t} \in \mathcal{X}_i$ at time $t = 0, \ldots, T$. The robot dynamics are:

$$x_{i,t+1} = f_i(x_{i,t}, u_{i,t}), \tag{3.9}$$

where $u_{i,t} \in \mathcal{U}_i$ is the control input and $\mathcal{U}_i$ is a finite set. We denote a control sequence as $\sigma_i = u_{i,0}, \ldots, u_{i,T-1} \in \mathcal{U}_i^T$.

The robots' goal is to track targets with state $y \in \mathbb{R}^{d_y}$ that have the following linear-Gaussian motion model:

$$y_{t+1} = A_t y_t + w_t, \quad w_t \sim \mathcal{N}(0, W_t), \tag{3.10}$$

where $A_t \in \mathbb{R}^{d_y \times d_y}$ and w_t is a zero-mean Gaussian noise with covariance $W_t \succeq 0$. The robots are equipped with sensors that measure the target state subject to an observation model:

$$z_{i,t} = H_{i,t}(x_{i,t}) y_t + v_{i,t}(x_{i,t}), \quad v_{i,t} \sim \mathcal{N}(0, V_{i,t}(x_{i,t})), \tag{3.11}$$

where $z_{i,t} \in \mathbb{R}^{d_{z_i}}$ is the measurement taken by robot i in state $x_{i,t}$, $H_{i,t}(x_{i,t}) \in \mathbb{R}^{d_{z_i} \times d_y}$, and $v_{i,t}(x_{i,t})$ is a state-dependent Gaussian noise, whose values are independent at any pair of times and across sensors. The observation model is linear in target states but can be nonlinear in the robot states. If it depends nonlinearly on the target state, we can linearize it around an estimate of the target state to get a linear model.

We assume every robot i has access to N_i control trajectories $\mathcal{M}_i = \{\sigma_i^k\}_{k=1}^{N_i}$ to choose from. Denote the set of all control trajectories as $\mathcal{M} = \cup_{i=1}^{n} \mathcal{M}_i$ and its size as $N = |\mathcal{M}|$. Potential control trajectories can be generated by various single-robot information gathering algorithms such as [2, 23, 79, 65]. The fact that every robot cannot execute more than one trajectory can be encoded as a partition matroid $(\mathcal{M}, \mathcal{I})$, where $\mathcal{M}$ is the ground set, and $\mathcal{I} = \{S \subseteq \mathcal{M} \mid |S \cap \mathcal{M}_i| \leq 1 \ \forall i \in \mathcal{R}\}$ consists of all admissible subsets of trajectories. Given $S \in \mathcal{I}$, we denote the joint state of robots that have been assigned trajectories as $x_{S,t}$ at time t, and their indices as $\mathcal{R}_S := \{i \mid |\mathcal{M}_i \cap S| = 1 \ \forall \ i \in \mathcal{R}\}$. Also, denote the measurements up to time $t \leq T$ collected by robots $i \in \mathcal{R}_S$ who follow the trajectories in S by $z_{S,1:t}$.

Due to the linear-Gaussian assumptions in (3.10) and (3.11), the optimal estimator for the target states is a Kalman filter. The target estimate covariance $\Sigma_{S,t}$ at time t resulting from robots $\mathcal{R}_S$ following trajectories in S obeys:

$$\Sigma_{S,t+1} = \rho^{\mathrm{e}}_{S,t+1}(\rho^{\mathrm{p}}_t(\Sigma_{S,t}), x_{S,t+1}), \tag{3.12}$$

where $\rho^{\mathrm{p}}_t(\cdot)$ and $\rho^{\mathrm{e}}_{S,t}(\cdot,\cdot)$ are the Kalman filter prediction and measurement update, respectively:

$$\begin{aligned}
\text{Predict:} \quad & \rho^{p}_t(\Sigma) := A_t \Sigma A_t^T + W_t, \\
\text{Update:} \quad & \rho^{\mathrm{e}}_{S,t}(\Sigma, x_{S,t}) := \left(\Sigma^{-1} + \sum_{i \in \mathcal{R}_S} M_{i,t}(x_{i,t}) \right)^{-1}, \\
& M_{i,t}(x_{i,t}) := H_{i,t}(x_{i,t}) V_{i,t}(x_{i,t})^{-1} H_{i,t}(x_{i,t})^T.
\end{aligned}$$

The robots are assumed to know each other's dynamics (3.9) and observation models (3.11)

so that they can assess the overall tracking performance given others' trajectories.

When choosing sensing trajectories, we want to capture the trade-off between sensing performance and energy expenditure, which is formalized below.

Problem 4 (Energy-Aware Active Information Acquisition) Given initial states $x_{i,0} \in \mathcal{X}_i$ for every robot $i \in \mathcal{R}$, a prior distribution of target state y_0, and a finite planning horizon T, find a set of trajectories $S \in \mathcal{M}$ to optimize the following:

$$\max_{S \in \mathcal{I}} J(S) := \mathbb{I}(y_{1:T}; z_{S,1:T}) - C(S), \tag{3.13}$$

where $\mathbb{I}(y_{1:T}; z_{S,1:T}) = \frac{1}{2}\sum_{t=1}^{T}\left[\log\det\left(\rho_{t-1}^{\mathrm{p}}(\Sigma_{S,t-1})\right) - \log\det(\Sigma_{S,t})\right] \geq 0$ is the mutual information between target states and observations, and the energy objective $C : 2^{\mathcal{M}} \to \mathbb{R}$ is defined as:

$$C(S) := \sum_{\sigma_i \in S} r_i C_i(\sigma_i), \tag{3.14}$$

where $r_i \geq 0$ is a tuning parameter, and $0 \leq C_i(\cdot) \leq c^{\max}$ is a non-negative, bounded energy cost for robot i to apply control sequence σ_i.

Remark 12 The optimization problem (3.13) is non-monotone, because adding extra trajectories may worsen the objective by incurring high energy cost $C(S)$. Thus, the constraint $S \in \mathcal{I}$ may not be tight, i.e.,, some robots may not get assigned trajectories. This property is useful when a large repository of heterogeneous robots is available but only a subset is necessary for the given tasks.

Remark 13 The choice of (3.13) is motivated by the energy-aware target tracking application. However, the proposed algorithm in Sec. C.3 is applicable to any scenario where $J(S)$ is a submodular set function that is not necessarily monotone, but can be made non-negative with proper offset.

Solving Problem 4 is challenging because adding energy cost $C(S)$ breaks the monotonicity of the objective, a property required for approximation methods such as coordinate descent [4] and the greedy algorithm [67] to maintain their performance guarantees. This is because these methods only add elements to the solution set, which always improves a monotone objective, but can worsen the objective in our setting, and may yield arbitrarily poor performance. We now propose a new distributed algorithm for Problem 4 based on local search [78].

C.3 Multi-Robot Planning

We first present how local search [78] can be used to solve Problem 4 with near-optimal performance guarantee. Despite the guarantee, local search is not suitable for distributed robot teams, because it assumes access to all locally planned robot control trajectories which can be communication-expensive to gather. To address this problem, we propose a new distributed algorithm that exploits the structure of a partition matroid to allow robots to collaboratively build a team plan by repeatedly proposing modifications to the collective trajectories. Moreover, we develop techniques to reduce computation and communication without affecting the performance guarantees.

In the following subsections, we denote $g : 2^{\mathcal{M}} \to \mathbb{R}$ as the non-negative, submodular oracle function used by local search, where the ground set $\mathcal{M}$ consists of the robot trajectories.

C.4 Centralized Local Search (`CLS`)

We present the original local search [78] in our setting with a single partition matroid constraint. We refer to it as centralized local search (`CLS`, Alg. 6) because it requires access to trajectories $\mathcal{M}$ from all robots. The algorithm proceeds in two rounds to find two candidate solutions $S_1, S_2 \in \mathcal{I}$. In each round $k = 1, 2$, solution S_k is initialized with a single-robot trajectory maximizing the objective (Line 5). Repeatedly, S_k is modified by

Algorithm 6 Centralized Local Search [78] (CLS)

1: **require** $\alpha > 0$, ground set $\mathcal{M}$, admissible subsets $\mathcal{I}$, oracle g
2: $N \leftarrow |\mathcal{M}|$
3: $S_1, S_2 \leftarrow \emptyset$
4: **for** $k = 1, 2$ **do**
5: $\quad S_k \leftarrow \{\text{argmax}_{a \in \mathcal{M}} g(\{a\})\}$ // Initialize with best traj.
6: $\quad$ **while** resultant S'_k from ①, ② or ③ satisfies $S'_k \in \mathcal{I}$ and $g(S'_k) \geq (1 + \frac{\alpha}{N^4}) g(S_k)$ **do** $S_k \leftarrow S'_k$ // Repeat local operations
7: $\quad\quad$ ① **Delete:** $S'_k \leftarrow S_k \backslash \{d\}$, where $d \in S_k$
8: $\quad\quad$ ② **Add:** $S'_k \leftarrow S_k \cup \{a\}$, where $a \in \mathcal{M} \backslash S_k$
9: $\quad\quad$ ③ **Swap:** $S'_k \leftarrow S_k \backslash \{d\} \cup \{a\}$, where $d \in S_k$, $a \in \mathcal{M} \backslash S_k$
10: $\quad \mathcal{M} \leftarrow \mathcal{M} \backslash S_k$
11: **return** $\text{argmax}_{S \in \{S_1, S_2\}} g(S)$

executing one of the Delete, Add or Swap operations, if it improves the objective by at least $(1 + \frac{\alpha}{N^4})$ of its original value (Lines 6–9), where $\alpha > 0$ controls run-time and performance guarantee. This procedure continues until S_k is no longer updated, and the next round begins without considering S_k in the ground set $\mathcal{M}$ (Line 10). Lastly, one of S_1 and S_2 that results in better objective is returned.

One important requirement of CLS is that the objective function g is non-negative. With the objective from Problem 4, this may not be true, therefore we add an offset O. The next proposition provides a worst-case performance guarantee for applying Alg. 6 to Problem 4 after properly offsetting the objective to be non-negative.

Proposition 14 Consider that we solve Problem 4 whose objective is made non-negative by adding a constant offset:

$$\max_{S \in \mathcal{I}} \; g(S) := J(S) + O, \tag{3.15}$$

where $O := \sum_{i=1}^{n} r_i c^{\max}$. Denote S^* and S^{ls} as the optimal solution and solution obtained by *CLS* (Alg. 6) for (3.15), by using $g(\cdot)$ as the oracle. We have the following worst-case performance guarantee for the objective:

$$0 \leq g(S^*) \leq 4(1 + \alpha) g(S^{\text{ls}}). \tag{3.16}$$

Proof 3 In (3.13), mutual information is a submodular set function defined on measurements provided by selected trajectories [4]. Furthermore, $C(S)$ is modular due to its additive nature:

$$C(S) = \sum_{\sigma_i \in S} r_i C_i(\sigma_i) \geq 0. \tag{3.17}$$

Since mutual information is non-negative, (3.15) is a submodular non-monotone maximization problem with a partition matroid constraint. The proposition follows from [78, Thm. 4].

Remark 15 Having the constant O term in (3.15) does not change the optimization in Problem 4, but ensures that the oracle used by *CLS* (Alg. 6) is non-negative so that the ratio $(1+\frac{\alpha}{N^4})$ correctly reflects the sufficient improvement condition.

Besides the communication aspect that CLS requires access to all robot trajectories, running it naively can incur significant computation. In the worst case, CLS requires $\mathcal{O}(\frac{1}{\alpha}N^6\log(N))$ oracle calls[†], where N is the total number of trajectories [78]. In practice, however, the run-time can be significantly reduced by our proposed distributed algorithm (see results in Sec. D), while maintaining a worst-case guarantee.

C.5 Distributed Local Search (DLS)

In this section, we propose a distributed implementation of the local search algorithm to address both the communication and computation concerns. The algorithm is presented in Alg. 7 and Alg. 8 written from robot i's perspective. Exploiting the structure of the partition matroid, DLS allows every robot to propose local operations based on its own trajectory set, while guaranteeing that the team solution never contains more than one trajectory for every robot. In fact, all steps executed by CLS can be distributedly proposed by robots, thus providing the same performance guarantee shown in Theorem 14.

[†]For 2 solution candidates, each requires $\mathcal{O}(\frac{1}{\alpha}N^4\log(N))$ local operations, and N^2 oracle calls to find each local operation in the worst case.

C.5.1 Distributed Proposal

Every proposal consists of two trajectories (d,a), where d is to be deleted from and a is to be added to the solution set. We also define a special symbol "`NOP`" that leads to no set operation, i.e.,, $S_k \cup \{\texttt{NOP}\} = S_k \backslash \{\texttt{NOP}\} = S_k$. Note that $(d,\texttt{NOP})$, $(\texttt{NOP},a)$ and (d,a) are equivalent to the Delete, Add and Swap steps in `CLS`.

Every robot i starts by sharing size of its trajectory set $|\mathcal{M}_i|$ and its best trajectory $a_i^* \in \mathcal{M}_i$ in order to initialize S_k and N collaboratively (Alg. 7 Lines 5–7). Repeatedly, every robot i executes the subroutine `FindProposal` (Alg. 8) in parallel, in order to propose changes to S_k (Alg. 7 Lines 8–13). Since any valid proposal shared by robots will improve the objective, the first $(d,a) \neq (\texttt{NOP},\texttt{NOP})$ will be adopted by every robot to update S_k in every round (Alg. 7 Lines 10–12). We assume instantaneous communication, so robots always use a common proposal to update their local copies of S_k. Otherwise, if delay leads to multiple valid proposals, a resolution scheme is required to ensure robots pick the same proposal.

In `FindProposal` (Alg. 8), an outer loop looks for potential deletion $d \in S_k$ (Alg. 8 Lines 2–6). Otherwise, further adding $a \in \mathcal{M}_i$ is considered, as long as the partition matroid constraint is not violated (Alg. 8 Lines 7–8). Next, we discuss how to efficiently search for trajectories to add.

C.5.2 Lazy Search

Instead of searching over trajectories in an arbitrary order, we can prioritize the ones that already perform well by themselves, based on $g(a|\emptyset)$ for all $a \in \mathcal{M}_i$ (Alg. 7 Line 2). In this fashion, we are more likely to find trajectories that provide sufficient improvement earlier (Alg. 8 Lines 12–13). Note that $g(a|\emptyset)$ is typically a byproduct of the trajectory generation process, so it can be saved and reused.

This ordering also allows us to prune unpromising trajectories. Given the team solution after deletion $S_k^- := S \backslash \{d\}$, the required marginal gain for subsequently adding trajectory a

Algorithm 7 Distributed Local Search (`DLS`)

require $\alpha > 0$, trajectories $\mathcal{M}_i$, oracle g
Sort $\mathcal{M}_i$ in descending order based on $g(a|\emptyset)$ for all $a \in \mathcal{M}_i$
$S_1, S_2 \leftarrow \emptyset$
for $k = 1, 2$ **do**
    Broadcast $|\mathcal{M}_i|$ and $a_i^* \in \mathcal{M}_i$ that maximizes $g(\{a_i^*\})$
    $S_k \leftarrow \{a^*\}$, where $a^* \in \{a_i^*\}_{i=1}^n$ maximizes $g(\{a^*\})$
    $N \leftarrow \sum_{i=1}^n |\mathcal{M}_i|$
    repeat
        Run `FindProposal`$(S_k, \mathcal{M}_i, \alpha, N, g)$ in background
        if Receive $(d, a) \neq (\texttt{NOP}, \texttt{NOP})$ **then**
            Terminate `FindProposal` if it has not finished
            $S_k \leftarrow S_k \backslash \{d\} \cup \{a\}$
    until Receive $(d, a) = (\texttt{NOP}, \texttt{NOP})$ from all robots
    $\mathcal{M}_i \leftarrow \mathcal{M}_i \backslash S_k$
return $\operatorname{argmin}_{S \in \{S_1, S_2\}} g(S)$

is

$$g(a|S_k^-) \geq \Delta := (1 + \frac{\alpha}{N^4}) g(S_k) - g(S_k^-). \tag{3.18}$$

We can prune any $a \in \mathcal{M}_i$, if $g(a|\emptyset) < \Delta$ based on the diminishing return property: because $\emptyset \subseteq S_k^-$, we know that $\Delta > g(a|\emptyset) \geq g(a|S_k^-)$, violating condition (3.18). Similarly, all subsequent trajectories a' can be ignored, because their marginal gains $g(a'|\emptyset) \leq g(a|\emptyset) < \Delta$ due to ordering (Alg. 8 Lines 10–11). Lastly, if adding a trajectory improves S_k^- sufficiently, the proposal is broadcasted (Alg. 8 Lines 12–13).

C.5.3 Greedy Warm Start

We observe empirically that a robot tends to swap its own trajectories consecutively for small growth in the objective, increasing communication unnecessarily. This can be mitigated by a simple technique: when finding local operations initially, we force robots to only propose additions to greedily maximize the objective, until doing so does not lead to enough improvement or violates the matroid constraint. Then robots resume normally with Alg. 8, allowing all local operations. By warm starting the team solution greedily, every robot

Algorithm 8 Find Proposal (`FindProposal`)

require S_k, $\mathcal{M}_i$, $\alpha > 0$, N, g
for $d \in S_k$ or $d = \texttt{NOP}$ **do** // Delete d, or no deletion
  $S_k^- \leftarrow S_k \backslash \{d\}$
  $\Delta \leftarrow (1 + \frac{\alpha}{N^4}) g(S_k) - g(S_k^-)$ // Δ: deficiency of S_k^-
  if $\Delta \leq 0$ **then**
    broadcast $(d, \texttt{NOP})$
  if $\exists\, a \in S_k^-$ planned by robot i **then**
    continue // Cannot add due to partition matroid
  for $a \in \mathcal{M}_i$ in sorted order **do** // Add a
    if $g(a|\emptyset) < \Delta$ **then**
      break // Rest of $a \in \mathcal{M}_i$ will not improve S_k^- enough
    if $g(a|S_k^-) \geq \Delta$ **then**
      broadcast (d, a)
broadcast $(\texttt{NOP}, \texttt{NOP})$

aggregates numerous proposals with smaller increase in the objective into a greedy addition with larger increase, thus effectively reducing communication.

D Simulation Results

We evaluate `DLS` in two target tracking scenarios based on objective values, computation, communication, and ability to handle heterogeneous robots. Its performance is compared against coordinate descent (`CD` [4]), a state-of-the-art algorithm for multi-robot target tracking that, however, assumes monotonicity of the objective. Planning for robots sequentially, `CD` allows every robot to incorporate the plans of previous robots. We also allow `CD` to not assign any trajectory to a robot if it worsens the objective. Reduced value iteration [2] is used to generate trajectories for both algorithms. Comparisons between `CLS` and `DLS` are omitted because the two algorithms empirically achieve the same average performance. We set $\alpha = 1$ arbitrarily, because tuning it was not effective due to the large number of trajectories N.

Both `DLS` and `CD` are implemented in C++ and evaluated in simulation on a laptop with an Intel Core i7 CPU. For `DLS`, every robot owns separate threads, and executes Alg. 8 over

4 extra threads to exploit its parallel structure. Similarly, `CD` allows every robot to use 4 threads and additionally incorporates accelerated greedy [80] for extra speed-up.

D.1 Robots Characteristics

Given initial state $x_{i,0} \in \mathcal{X}_i$ for robot $i \in \mathcal{R}_S$ who follows the control sequence $u_{i,0}, \ldots, u_{i,T-1} = \sigma_i \in S$, the resultant states are $x_{i,1}, \ldots, x_{i,T}$ based on dynamics (3.9). The energy cost $C(S)$ may also be state-dependent. We define it as:

$$C(S) := \sum_{i \in \mathcal{R}_S} r_i \sum_{t=0}^{T-1} \left(c_i^{\text{ctrl}}(u_{i,t}) + c_i^{\text{state}}(x_{i,t}) \right), \tag{3.19}$$

where the state-dependent cost $c_i^{\text{state}}(\cdot)$ and control-dependent cost $c_i^{\text{ctrl}}(\cdot)$ are defined based on robot types—in our case, robot i is either an unmanned ground vehicle (UGV) or an unmanned aerial vehicle (UAV). Note that decomposition between state and control is not required for our framework to work. The setup for robots are summarized in Table 3.1. For simplicity, all robots follow differential-drive dynamics[‡] with sampling period $\tau = 0.5$ and motion primitives consisting of linear and angular velocities $\{u = (v, \omega) \mid v \in \{0, 8\}$ m/s, $\omega \in \{0, \pm\frac{\pi}{2}\}$ rad/s$\}$. We consider muddy and windy regions that incur state-dependent costs for UGVs and UAVs, respectively. The robots are equipped with range and bearing sensors, whose measurement noise covariances grow linearly with target distance. The sensors have limited ranges and field of views (FOVs), within which the maximum noise standard deviations are 0.1 m and 5° for range and bearing measurements, respectively. Outside the range or field of view, measurement noise becomes infinite. Refer to [1] for details.

[‡]We note that any dynamically feasible model can be used for the specific robot which is being planned for. We use the same kinematic model for the quadrotor and ground vehicle for implementation convenience, and because the quadrotors are restricted to a plane to avoid collisions.

D.2 Scenario 1: Multi-Robot Dynamic Target Tracking

Here we show the computation and communication savings for DLS, and compare the performance of DLS and CD (see Fig. 3.6 and 3.7). The scenario involves $2, \ldots, 10$ UGVs trying to estimate the positions and velocities of the same number of dynamic targets that follow discretized double integrator dynamics. Targets have a top speed of 2 m/s and their models are corrupted by Gaussian noise. Robots and targets are spawned in a square arena whose sides grow from 40 m to 60 m, and 50 random trials are run for each number of robots.

Non-monotonicity in the problem is accentuated by an increasing penalty for control effort of additional robots, by setting $r_i = i$ for each robot i as defined in (3.19) (i.e.,, the 10-th added robot is 10 times more expensive to move than the first). Note that state-dependent cost is set to 0 only for this experiment. Trajectory generation has parameters $\varepsilon = 1$ and $\delta = 2$ for horizon $T = 10$. As the planning order is arbitrary for CD, we investigate two planning orders: first from cheaper to more expensive robots, and the reverse. Intuitively and shown in Fig. 3.7, the former should perform better than the later, because the same amount of information can be gathered while spending less energy. While other orderings are possible (e.g.,,[63] uses CD to decentralize greedy algorithm), we only use two to show CD's susceptibility to poor planning order. To ensure a fair comparison between DLS and CD, we use a fixed set of trajectories generated offline, but ideally trajectories should be replanned online for dynamic target tracking.

	$c^{\text{ctrl}}(u)$, u given as			$c^{\text{state}}(x)$, x in		FOV (°)	Range (m)
	$0, 0$	$0, \frac{\pm\pi}{2}$	$8, \frac{\pm\pi}{2}$	Mud	Wind	Exp.1&2	Exp.1&2
UGV	0	1	2	3	/	160	6 & 15
UAV	2	2	4	/	3	360	/ & 20

Table 3.1: Robot setup in two experimental scenarios.

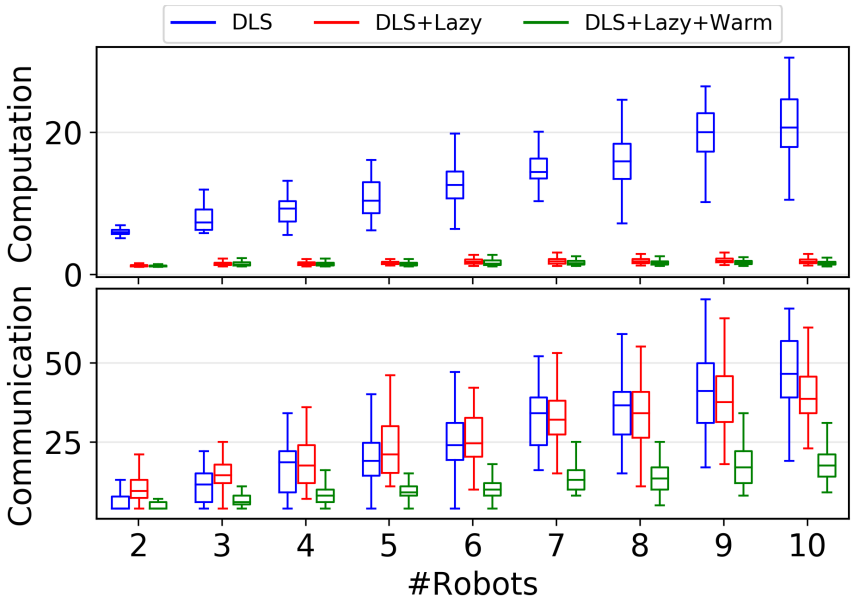


Figure 3.6. Computation and communication savings afforded by lazy search (*Lazy*) and greedy warm start (*Warm*) for DLS. Computation is measured by total oracle calls divided by the number of trajectories N, where N reaches around 12500 for 10 robots. Communication is measured by the number of proposal exchanges. Combining lazy search and greedy warm start (green) leads to 80–92% computation reduction, and up to 60% communication reduction compared to the naive implementation (blue) on average.

Proposed methods for improving naive distributed execution of local search, namely lazy search (Lazy) and greedy warm start (Warm), are shown to reduce computation by 80–92% and communication by up to 60% on average, as shown in Fig. 3.6. As expected, when there are few robots with similar control penalties, the objective is still close to being monotone and DLS and CD achieve similar performance as shown in Fig. 3.7. However, as more costly robots are added, their contribution in information gain is reduced by high control penalty, the problem becomes more non-monotone. Therefore, the performance gap between DLS and CD widens, because CD requires monotonicity to maintain its performance guarantee, but DLS does not. From Fig. 3.7, we can see that planning order is critical for CD to perform well, but a good ordering is often unknown a priori. Compared to CD which requires only $n-1$ communication rounds for n robots, DLS requires more for its performance. For practical concerns to save more time, DLS with down-sampled trajectories

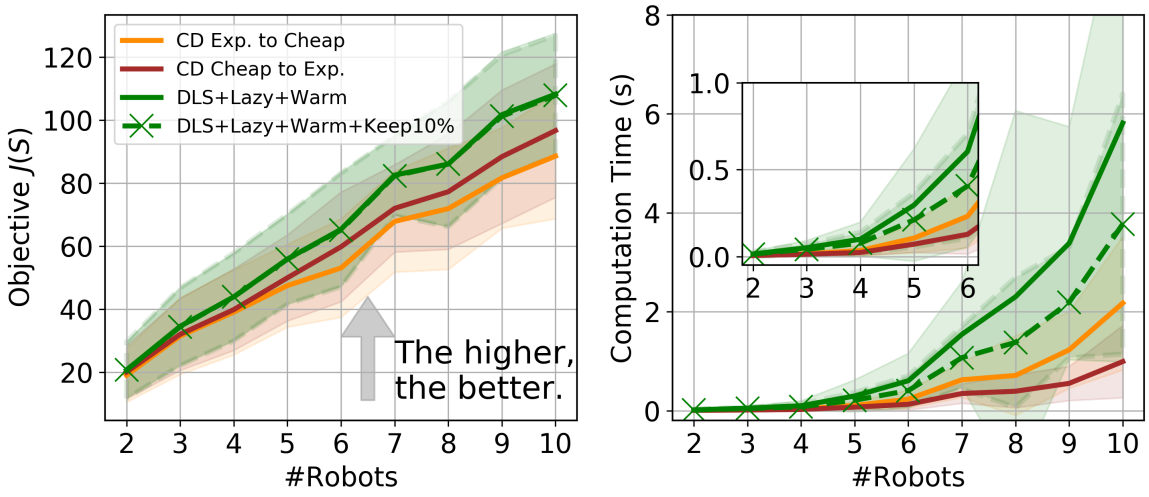


Figure 3.7. Objective values and computation time (s) for variants of DLS and CD, where the lines and shaded areas show the mean and standard deviation, respectively. The time excludes the trajectory generation time (< 2 s), which is the same for every algorithm. DLS (solid green) consistently outperforms CD in optimizing the objective, where it is better for CD to plan from cheaper to more expensive robots (brown), rather than the reverse order (orange). The performance gap between DLS and CD widens as more costly robots increase non-monotonicity of the problem. However, DLS requires longer run-time, which in practice can be alleviated by using a portion of all trajectories. This invalidates the worst-case guarantee, but DLS solution based on the best 10% of each robot's trajectories (green crosses) still outperforms CD.

(e.g.,, keeping the best 10% of each robot's trajectories) still produces better solution than CD, but the guarantee of DLS no longer holds.

D.3 Scenario 2: Heterogeneous Sensing and Control

Here we consider a heterogeneous team consisting of 2 UGVs and 1 UAV with different sensing and control profiles (Table 3.1) in order to track 10 static targets in a 100 m $\times$ 100 m arena, over a longer horizon $T = 20$ (see Fig. 3.9). The UAV has better sensing range and field of view compared to UGVs, but consumes more energy. The arena has overlapping muddy and windy regions, so robots have to collaboratively decide who should venture into the costly regions. To explore trade-off between sensing and energy objectives as a team, we set a common $r_i = r$ for every robot i. We vary r from 0 to 0.5 and run 50 trials for each fixed value. Robots are spawned in the non-muddy, non-windy region, but targets may

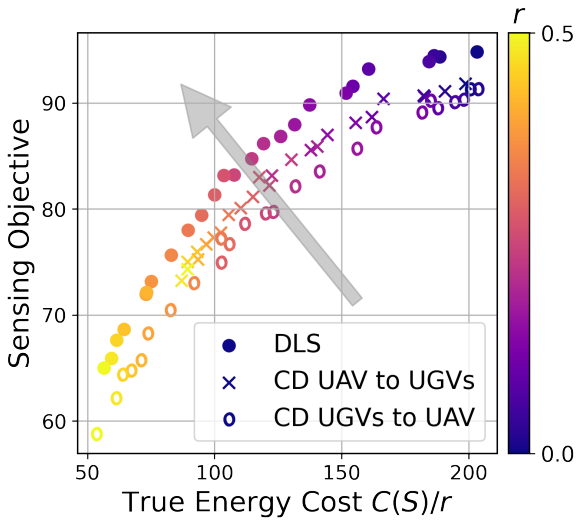


Figure 3.8. Trade-off between sensing performance (mutual information (3.13)) and the true energy expenditure $C(S)/r$ in heterogeneous robot experiments produced by DLS and CD, where it is better to be in the upper left. Each point is an average obtained over 50 trials for a fixed r, where we set $r_i = r$ for each robot i to penalize the team energy expenditure per (3.19).

appear anywhere. Like before, we evaluate two CD planning orders: from UAV to UGVs, and the reverse. Different from last scenario, we set $\delta = 4$ to handle the longer horizon.

As shown in Fig. 3.8, DLS consistently achieves better sensing and energy trade-off than CD on average. To gain intuitions on why CD under-performs, a particular trial given $r = 0.2$ is shown in Fig. 3.9. Due to the non-monotone objective, the robot who plans first to maximize its own objective can hinder robots who plan later, thus negatively affecting team performance.

E Conclusion

This chapter considered multi-robot information gathering problems, and proposed two algorithms for solving them, namely coordinate descent (CD), and distributed local search (DLS). Coordinate descent provides constant factor performance guarantees for problems that are monotone, and submodular. We then presented simulations and hardware experiments which operate in real time, leveraging the single robot planner ARVI, presented in

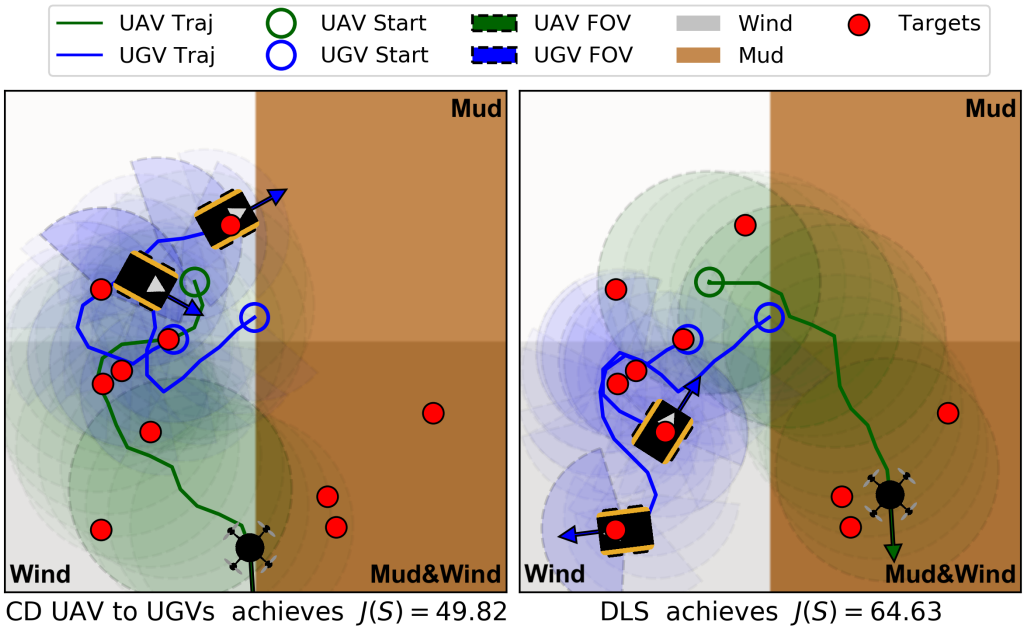


Figure 3.9. Example solutions from `CD` (left) and `DLS` (right) for 2 UGVs and 1 UAV with $r = 0.2$ that penalizes energy cost $C(S)$ in (3.19). The arena is both windy and muddy, which is costly for the UAV and UGVs, respectively. (Left) `CD` performs poorly due to its fixed planning order: the UAV plans first to hover near the targets on the left, rather than venturing over the mud. Thus, the UGVs are under-utilized because they are unwilling to go into the mud to observe the targets on the bottom right. For similar reasons, `CD` with reversed order under-utilizes the UAV, which is not visualized due to limited space. (Right) In contrast, `DLS` deploys the UAV over the muddy regions, leading to a better value of $J(S)$ in (3.13).

Chapter 1. The DLS algorithm is adapted to solve a novel energy-aware information acquisition problem, which is non-monotone. The algorithm is well suited to heterogeneous teams, and we present simulation results indicating DLS performs better in non-monotone problems than Coordinate Descent.

Chapter 4

Resilient Information Acquisition

A Introduction

Until this point, most robotics research including the work developed in this dissertation places no emphasis on robustifying information acquisition against attacks (or failures) that may temporarily disable the robots' sensors; e.g., smoke-cloud attacks that can block temporarily the field of view of multiple sensors. For example, [29] focuses on formation control, instead of information acquisition; [81] focuses on state estimation against byzantine attacks, i.e., attacks that corrupt the robots' sensors with adversarial noise, instead of disabling them; and [82] focuses on a trajectory scheduling in transportation networks when travel times can be uncertain, instead on trajectory planning for information acquisition. An exception is [26], which however is limited to multi-target tracking based on myopic planning, instead of non-myopic.

Related work has also been developed in combinatorial optimization [21, 83], paving the way for robust combinatorial optimization against attacks: [84] proposes algorithms for submodular set function optimization against worst-case attacks, but under the assumption the attacks can remove (disable) only a limited number of elements from the optimized set. Instead, [85] proposes algorithms for optimization against any number of removals. And,

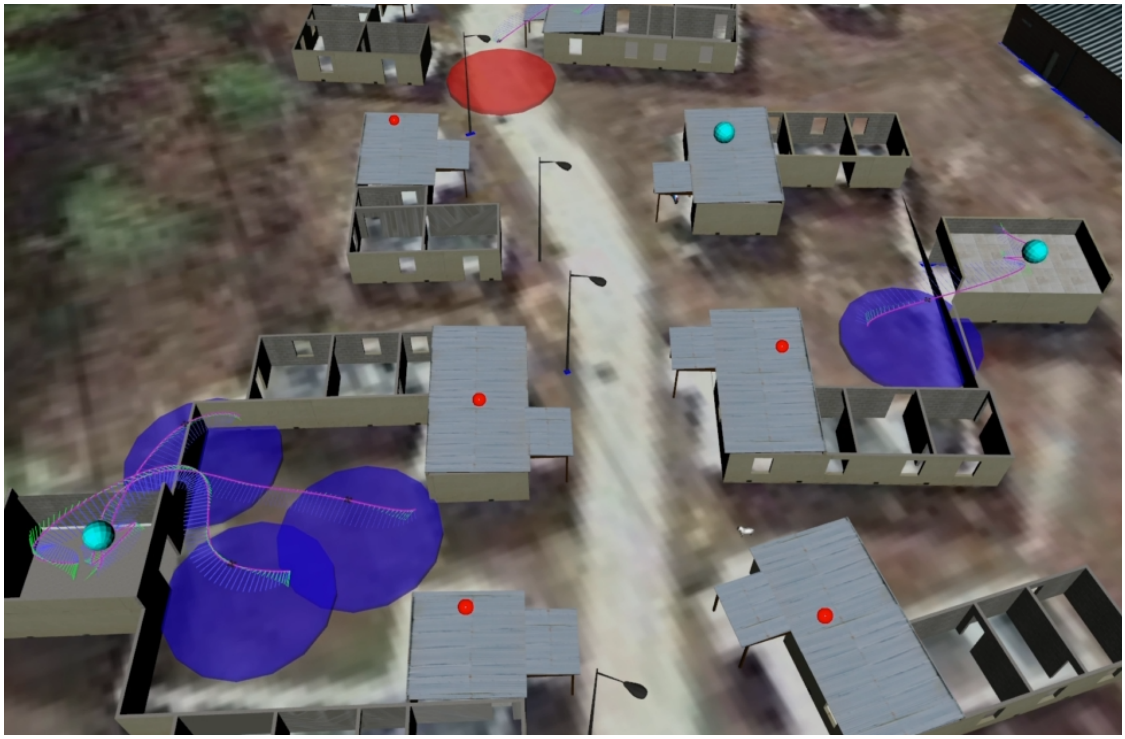


Figure 4.1. **Persistent Surveillance under Attacks.** Unity simulation environment depicting a 5-robot team engaging in a *Persistent Surveillance* task for monitoring a set of buildings. Some robots are under attack. The attacks can disable the sensing capabilities of the robots, at least temporarily. Each blue disc indicates the field of view of a (non-attacked) robot, while each red disc indicates an attacked robot. In this adversarial environment, the robots must resiliently plan trajectories to (re-)visit all the building landmarks to continue acquiring information despite the attacks.

recently, [86] extended the algorithms to matroid constraints, enabling the application of the algorithms in robotics since multi-robot path planning can be cast as a matroid-constrained optimization problem [5].

Contributions. In this chapter, in contrast to the aforementioned works, we formulate a new resilient information acquisition problem that accounts for attacks against the robots while simultaneously performing non-myopic multi-robot planning. We make the following three key contributions.

1. Receding Horizon Formulation and Algorithm. Section B formalizes the attack-aware active information acquisition problem as a finite horizon control optimization problem named Resilient Active Information acquisitioN (P-RAIN) —in the acronym, the "P" stands for "Problem." (P-RAIN) is a sequential mixed-integer optimization problem: it jointly optimizes the robots' control inputs such that the robots' planned paths are robust against worst-case attacks that may occur at each time step. The upper-bound to the number of attacks is assumed known and constant. (P-RAIN)'s information objective function is assumed non-decreasing in the number of robots (a natural assumption, since the more robots the more information one typically can collect).

Section C proposes RAIN , the first receding horizon algorithm for the problem of resilient active information acquisition (P-RAIN). RAIN calls in an online fashion Robust Trajectory Planning (RTP), a subroutine that plans attack-robust control inputs over a planning horizon. RTP 's planning horizon is typically less than the acquisition problem's finite horizon, for computational reasons. For the same reason, RTP assumes constant attacks. RTP is presented in Section D.

2. Performance Analysis. Although no performance guarantees exist for the non-linear combinatorial optimization problem (P-RAIN), Section E provides suboptimality bounds on RTP 's performance, i.e., on the algorithm used by RAIN to approximate a solution to (P-RAIN) locally, in a receding horizon fashion. The theoretical analysis is based on notions of curvature introduced for combinatorial optimization; namely, the notions of curvature [87]

and total curvature [88]. The notions aim to bound the worst-case complementarity of the robots' planned paths in their ability to jointly maximize (P-RAIN) information acquisition objective function.

3. Experiments. Section F evaluates RAIN across three multi-robot information acquisition tasks: Multi-Target Tracking, Occupancy Grid Mapping, and Persistent Surveillance. All evaluations demonstrate the necessity for attack-resilient planning, via a comparison with a state-of-the-art baseline information acquisition algorithm, namely, coordinate descent [4]. Specifically, RAIN runs in real-time and exhibits superior performance in all experiments. RAIN 's effectiveness is accentuated the higher the numbers of attacks is (Section F.1). RAIN remains effective even against non-worst-case attacks, specifically, random (Section F.2). Even when high replanning rates are feasible (Section F.3), in which case coordinate descent can adapt at each time step against the observed attacks, RAIN still exhibits superior performance. The algorithm is implemented in C++ and a Unity-based simulator.

t

B Resilient Active Information acquisitioN (RAIN) Problem

We present the optimization problem of Resilient Active Information acquisitioN (RAIN) (Section B.2). To this end, we first formalize the (attack-free) active information acquisition problem (Section B.1). We also use the notation:

- $\phi_{\mathcal{V},\,\tau:\tau+\tau'} \triangleq \{\phi_{i,t}\}_{i\in\mathcal{V},\,t\in[\tau,\ldots,\tau+\tau']}$, for any variable of the form $\phi_{i,t}$, where $\mathcal{V}$ denotes a set of robots (i.e., $i\in\mathcal{V}$ is robot i), and $[\tau,\ldots,\tau+\tau']$ denotes a discrete time interval ($\tau\geq 1$, while $\tau'\geq 0$);
- $w\sim\mathsf{N}(\mu,\Sigma)$ denotes a Gaussian random variable w with mean μ and covariance Σ.

B.1 Active Information Acquisition in the Absence of Attacks

Active information acquisition is a control input optimization problem over a finite-length time horizon: it looks to jointly optimize the control inputs for a team of mobile robots so that the robots, acting as mobile sensors, maximize the acquired information about a target process. Evidently, the optimization must account for the (a) robot dynamics, (b) target process, (c) sensor model, (d) robots' communication network, and (e) information acquisition objective function:

Robot Dynamics We assume noise-less, non-linear robot dynamics, adopting the framework introduced in [4]:

$$x_{i,t} = f_i(x_{i,t-1},\, u_{i,t-1}), \quad i \in \mathcal{V}, \quad t = 1,2,\ldots, \tag{4.1}$$

where $\mathcal{V}$ denotes the set of available robots, $x_{i,t} \in \mathbb{R}^{n_{x_{i,t}}}$ denotes the state of robot i at time t,* and $u_{i,t} \in \mathcal{U}_{i,t}$ denotes the control input to robot i; $\mathcal{U}_{i,t}$ denotes the finite set of admissible control inputs to the robot.

Target Process We assume any target process

$$y_t = g(y_{t-1}) + w_t, \quad t = 1,2,\ldots, \tag{4.2}$$

where y_t denotes the target's state at time t, and w_t denotes gaussian process noise; we consider $w_t \sim \mathsf{N}(\mu_{w_t}, \Sigma_{w_t})$.

Sensor Model We assume measurements of the form

$$z_{i,t} = h_i(x_{i,t},\, y_t) + v_{i,t}(x_{i,t}), \quad t = 1,2,\ldots, \tag{4.3}$$

*$x_{i,t}$ in eq. (4.1) belongs to an appropriate state space, such as $SE(2)$ or $SE(3)$, depending on the problem instance.

where $z_{i,t}$ denotes the measurement by robot i at time t, and $v_{i,t}$ denotes measurement noise; we consider $v_{i,t} \sim \mathsf{N}(\mu_{v_{i,t}}((x_{i,t}), \Sigma_{v_{i,t}}((x_{i,t}))$. Both the noise and sensor function h_i depend on $x_{i,t}$, as it naturally is the case for, e.g., bearing and range measurements (cf. Section F.1).

Communication Network among Robots We assume centralized communication, i.e., all robots can communicate with each other at any time.

Information Acquisition Objective Function The information acquisition objective function captures the acquired information about the target process, as collected by the robots during the task via their measurements. In this paper, in particular, we consider objective functions of the additive form

$$J_{\mathcal{V},\,1:\mathsf{T}_{\mathsf{TASK}}} \triangleq \sum_{t=1}^{\mathsf{T}_{\mathsf{TASK}}} J(y_t \mid u_{\mathcal{V},1:t},\, z_{\mathcal{V},1:t}), \tag{4.4}$$

where $\mathsf{T}_{\mathsf{TASK}}$ denotes the duration of the information acquisition task, and $J(y_t \mid u_{\mathcal{V},1:t},\, z_{\mathcal{V},1:t})$ is an information metric such as the conditional entropy [4] (also, cf. Section F.1) or the mutual information [5] (also, cf. Section F.2), where we make explicit only the metric's dependence on $u_{\mathcal{V},1:t}$ and $z_{\mathcal{V},1:t}$ (and we make implicit the metric's dependence on the initial conditions y_0 and $x_{i,0}$, and on the noise parameters, i.e., the means and covariances of w_t and v_t for $t = 1, \ldots, \mathsf{T}_{\mathsf{TASK}}$).

Problem 5 ((Attack-Free) Active Information Acquisition) At time $t = 0$, find control inputs $u_{\mathcal{V},\,1:\mathsf{T}_{\mathsf{TASK}}}$ by solving the optimization problem

$$\max_{\substack{u_{\mathcal{V},t} \in \mathcal{U}_{\mathcal{V},t} \\ t = 1:\mathsf{T}_{\mathsf{TASK}}}} \quad J_{\mathcal{V},\,1:\mathsf{T}_{\mathsf{TASK}}}. \tag{4.5}$$

Eq. (4.5) captures a control input optimization problem where across a task-length horizon, the control inputs of all robots are jointly optimized to maximize $J_{\mathcal{V},\,1:\mathsf{T}_{\mathsf{TASK}}}$.

Solving eq. (4.5) can be challenging, mainly due to (i) the non-linearity of eqs. (4.1)-(4.3), (ii) the duration of the task, $\mathsf{T}_{\mathsf{TASK}}$, which acts as a look-ahead planning horizon (the longer

the planning horizon is, the heavier eq. (4.5) is in computing an optimal solution), and (iii) that at $t = 0$ no measurements have been realized yet.

To overcome the aforementioned challenges, on-line solutions to eq. (4.5) have been proposed [4], similar to the Receding Horizon Control solution —also known as Model Predictive Control (MPC)— for the finite-horizon optimal control problem [89, Chapter 12]. Specifically, per the receding horizon approach, one aims to solve eq. (4.5) sequentially in time, by solving at each $t = 1, \ldots, \mathsf{T}_{\mathsf{TASK}}$ an easier version of eq. (4.5), but of the same form as eq. (4.5), where (i) the look-ahead horizon $\mathsf{T}_{\mathsf{TASK}}$ is replaced by a shorter $\mathsf{T}_{\mathsf{PLAN}}$ ($\mathsf{T}_{\mathsf{PLAN}} \leq \mathsf{T}_{\mathsf{TASK}}$), and (ii) eqs. (4.1)-(4.3) are replaced by their li- nearizations given the current $x_{\mathcal{V},t}$ and current estimate of y_t.

B.2 Active Information Acquisition in the Presence of Attacks

Eq. (4.5) may suffer, however, from an additional challenge: the presence of attacks against the robots, which, if left unaccounted, can compromise the effectiveness of any robot plans per eq. (4.5). In this paper, in particular, we consider the presence of the following type of attacks:

Attack Model At each t, an attack $\mathcal{A}_t$ can remove at most α robots from the information acquisition task ($\mathcal{A}_t \subseteq \mathcal{V}$ and $|\mathcal{A}_t| \leq \alpha$), in the sense that any removed robot i ($i \in \mathcal{A}_t$) cannot acquire any measurement $z_{i,t}$. In selecting the attack, the attacker has perfect knowledge of the state of the system. The attacker can select the worst-case attack (cf. Problem 1). Nevertheless, the attacker cannot necessarily prevent the robots from moving according to their pre-planned path, nor can cause any communication loss among the robots.

Problem 6 (Problem of Resilient Active Information acquisitioN (P-RAIN)) At time $t = 0$, find control inputs $u_{\mathcal{V},\,1:\mathsf{T}_{\mathsf{TASK}}}$ by solving the optimization problem

$$\max_{\substack{u_{\mathcal{V},t} \in \mathcal{U}_{\mathcal{V},t} \\ t=1:\mathsf{T}_{\mathsf{TASK}}}} \; \min_{\substack{\mathcal{A}_t \subseteq \mathcal{V},\, |\mathcal{A}_t| \leq \alpha \\ t=1:\mathsf{T}_{\mathsf{TASK}}}} \; J_{\mathcal{V}\setminus\mathcal{A}_t,\,1:\mathsf{T}_{\mathsf{TASK}}} \tag{P-RAIN}$$

(P-RAIN) goes beyond eq. (4.5) by accounting for the attacks $\mathcal{A}_t$ ($t = 1, \ldots, \mathsf{T}_{\mathsf{TASK}}$). This is expressed in (P-RAIN) with the minimization step, which aims to prepare an optimal solution $u_{\mathcal{V},t}$ against any worst-case attack that may happen at t.

Remark 1 (Need for (P-RAIN)**)** Reformulating the (attack-free) eq. (4.5) as in (P-RAIN) may seem unnecessary, since we consider that the attacker cannot cause any communication loss among the non-attacked robots (cf. the attack model defined above): indeed, if the non-attacked robots can instantaneously observe the attacks at each t, and instantaneously replan at the same moment t, then (P-RAIN) is unnecessary. However, replanning instantaneously in practice is impossible, due to (i) computationally-induced and algorithmic delays [4], as well as (ii) delays induced by the temporal discretization of the robot and target dynamics. Thus, for the duration replanning is impossible, the plans need to account for attacks.

C Receding Horizon Approximation: RAIN Algorithm

In solving (P-RAIN), one has to overcome not only the challenges involved in eq. (4.5) (cf. Section B) bult also the additional challenge of the worst-case attacks $\mathcal{A}_t$ (which are unknown a priori). We develop an on-line approximation procedure for (P-RAIN), summarized in Algorithm 9.

Intuitive Description. RAIN proposes a receding horizon solution to (P-RAIN), that enables on-line reaction to the history of attacks, and, thus, is resilient, by executing the steps:

Initialization ***(line 2)*** At $t = 0$, the acquisition task has not started and no attacks are assumed possible ($\mathcal{A}_0 = \emptyset$).

Receding Horizon Planning ***(lines 3-15)*** At each $t = 1, \ldots, \mathsf{T}_{\mathsf{TASK}}$, RAIN executes the receding horizon steps:

- Robust Trajectory Planning (RTP) (lines 4-7): Given the current estimate $\hat{y}_t$ of the target process, all robots jointly optimize their control inputs by solving Problem 7, presented next, which is of the same form as (P-RAIN) but where (i) the look-ahead horizon $\mathsf{T}_{\mathsf{TASK}}$ is replaced by a shorter $\mathsf{T}_{\mathsf{PLAN}}$ ($\mathsf{T}_{\mathsf{PLAN}} \leq \mathsf{T}_{\mathsf{TASK}}$), and (ii) the attack is considered fixed over the look-ahead horizon:

  Problem 7 (Robust Trajectory Planning (RTP)) At time t, find attack-robust control inputs $u_{\mathcal{V},\,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}$ by solving the optimization problem

$$\max_{\substack{u_{\mathcal{V},t'} \in \mathcal{U}_{\mathcal{V},t'} \\ t'=t+1:t+\mathsf{T}_{\mathsf{PLAN}}}} \quad \min_{\mathcal{A} \subseteq \mathcal{V},\, |\mathcal{A}| \leq \alpha} \quad J_{\mathcal{V}\setminus\mathcal{A},\,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}. \tag{P-RTP}$$

  Both aforementioned (i) and (ii) intend to make (P-RTP) computationally tractable, so (P-RTP) can be solved in real time for the purposes of receding horizon planning. Particularly, we assume that the the algorithm we propose for (P-RTP), RTP , is called in RAIN every $\mathsf{T}_{\mathsf{REPLAN}}$ steps.[†]

  Remark 2 (Role of $\mathsf{T}_{\mathsf{REPLAN}}$) $\mathsf{T}_{\mathsf{REPLAN}}$ is chosen so that a receding horizon plan can always be generated in the duration it takes to compute a solution to (P-RTP) via RTP ; e.g., if one timestep —real-time interval from any t to $t+1$— has duration $0.5s$, and solving (P-RTP) via RTP requires $2s$, then $\mathsf{T}_{\mathsf{REPLAN}} = 4$ steps. Generally, $\mathsf{T}_{\mathsf{REPLAN}} \geq 1$ steps. Factors that influence the required time to solve (P-RTP) include the size of the robot team, the length of the planning horizon $\mathsf{T}_{\mathsf{PLAN}}$ [2], the need for linearization of eqs. (4.1)-(4.3), and the number of possible attacks α —evidently, the latter factor is unique to (P-RAIN), in comparison to the attack-free eq. (4.5).

- Control Execution (lines 9-10): Each robot i uses their computed $u_{i,t}$ to make the next step in the environment (in the meantime, the real time changes from t to $t+1$ by the

[†]RTP's pseudo-code is presented in Algorithm 10, and described in more detail Section D. We quantify RTP's performance guarantees in Section E.

completion of the step).

- Attack Observation (line 11): RAIN observes the current attack, which affects the robots while they execute $u_{i,t}$.
- Measurement Collection (lines 12-13): The measurements from all non-attacked robots are collected.
- Estimate Update (line 14): Given all received measurements up to the current time, the estimate of y_t is updated.
- Time Update (line 15): RAIN updates the time counter to match it with the real time.

D Robust Trajectory Planning (RTP) Algorithm

We present RTP, which is used as a subroutine in RAIN, in a receding horizon fashion (cf. Section C). RTP's pseudo-code is presented in Algorithm 10. RTP's performance is quantified in Section E. We next give an intuitive description of RTP.

Intuitive Description. RTP's goal is to maximize (P-RTP)'s objective function $J_{\mathcal{V}\setminus\mathcal{A},\,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}$, despite a worst-case attack $\mathcal{A}$ that removes up to α robots from $\mathcal{V}$. In this context, RTP aims to fulfill (P-RTP)'s goal with a two-step process, where (i) RTP partitions robots into two sets (the set of robots $\mathcal{L}$, and the set of robots $\mathcal{V}\setminus\mathcal{L}$; cf. RTP's lines 1-3), and, then, (ii) RTP appropriately selects the robots' control inputs in each of the two sets (cf. RTP's lines 4-6). In particular, RTP picks $\mathcal{L}$ aiming to guess the worst-case removal of α robots from $\mathcal{V}$, i.e., to guess the optimal solution to the minimization step in (P-RTP). Thus, intuitively, $\mathcal{L}$ is aimed to act as a "bait" to the attacker. Since guessing the optimal "bait" is, in general, intractable [90], RTP aims to approximate it by letting $\mathcal{L}$ be the set of α robots with the α largest marginal contributions to $J_{\cdot,\,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}$ (RTP's lines 4-5). Then, RTP

Algorithm 9 Resilient Active Information acquisitioN (RAIN).

Require: RAIN receives the inputs:

- *Offline*: Duration $\mathsf{T}_{\mathsf{TASK}}$ of information acquisition task; look-ahead horizon $\mathsf{T}_{\mathsf{PLAN}}$ for planning trajectories ($\mathsf{T}_{\mathsf{PLAN}} \leq \mathsf{T}_{\mathsf{TASK}}$); replanning rate $\mathsf{T}_{\mathsf{REPLAN}}$ ($\mathsf{T}_{\mathsf{REPLAN}} \leq \mathsf{T}_{\mathsf{PLAN}}$); model dynamics f_i of each robot i's state $x_{i,t}$, including initial condition $x_{i,0}$ ($i \in \mathcal{V}$); sensing model h_i of each robot i's sensors, including $\mu_{v_{i,t}}$ and $\Sigma_{v_{i,t}}$; model dynamics g of target process, including initial condition y_0, and μ_{w_t} and covariance Σ_{w_t}; objective function J; number of attacks α.
- *Online*: At each $t = 1, \ldots, \mathsf{T}_{\mathsf{TASK}}$, observed (i) attack $\mathcal{A}_t$ (*i.e.*, robot removal $\mathcal{A}_t \subseteq \mathcal{V}$), and (ii) measurements $z_{i,t}$ from each non-attacked robot $i \in \mathcal{V} \setminus \mathcal{A}_t$.

At each $t = 1, \ldots, \mathsf{T}_{\mathsf{TASK}}$, estimate $\hat{y}_t$ of y_t.
// Initialize //
$t = 0$; $\hat{y}_t = y_0$; $\mathcal{A}_t = \emptyset$; $z_t = \emptyset$;
// Execute resilient active information acquisition task //
while $t < \mathsf{T}_{\mathsf{TASK}}$ **do**
//(Re)plan robust trajectories for all robots //
  if $t \mod \mathsf{T}_{\mathsf{REPLAN}} = 0$ **then**
    $\mathcal{I}_t = \{t, \{f_i, x_{i,t}, h_i, \mu_{w_t}, \Sigma_{w_t}, \mu_{v_{i,t}}, \Sigma_{v_{i,t}}\}_{i \in \mathcal{V}}, g, \hat{y}_t,$
      $z_t, \mathsf{T}_{\mathsf{PLAN}}, \alpha\}$; // Denote by $\mathcal{I}_t$ the information needed by the RTP algorithm, called in the next line.
    $u_{\mathcal{V},\, t+1:t} = \mathsf{RTP}(\mathcal{I}_{0:t})$
    // Plan robust trajectories for all robots with look-ahead planning horizon $\mathsf{T}_{\mathsf{PLAN}}$.
    // Execute current step of trajectory computed by RTP //
  for all $i \in \mathcal{V}$ **do**
    $x_{i,t+1} = f_i(x_{i,t}, u_{i,t})$;
  Observe $\mathcal{A}_{t+1}$;
  // Determined by environment/attacker.
 // Integrate measurements from non-attacked robots //
  for all $i \in \mathcal{V} \setminus \mathcal{A}_{t+1}$ **do**
    Receive measurement $z_{i,t+1}$; // Only measurements from non-attacked robots are received.
  update Estimate $\hat{y}_{t+1}$ of y_{t+1} given $z_{1:t+1}$; // $z_{1:t}$ collects all available measurements up to the time t, *i.e.*, $z_{1:t} \triangleq \{z_{i,\tau} : i \in \mathcal{V} \setminus \mathcal{A}_\tau, \tau = 1, \ldots, t\}$.
  $t = t + 1$; // Time update.

Algorithm 10 Robust Trajectory Planning (RTP).

Require: Look-ahead horizon $\mathsf{T}_{\mathsf{PLAN}}$ for planning trajectories; current time t; set of robots $\mathcal{V}$; model dynamics f_i of each robot i's state, including current state $x_{i,t}$; sensing model h_i of each robot i's sensors, including $\mu_{v_{i,t}}$ and $\Sigma_{v_{i,t}}$; model dynamics g of target process y_t, including current estimate $\hat{y}_t$, and μ_{w_t} and covariance Σ_{w_t}; objective function J; measurement history $z_{1:t}$; number of attacks α.

Ensure: Control inputs $u_{i,t'}$, for all robots $i \in \mathcal{V}$ and all times $t' = t+1, \ldots, t+\mathsf{T}_{\mathsf{PLAN}}$.

// Step 1: Generate *bait* robot set (to approximate a worst-case attack assumed constant $\forall t' \in [t+1, t+\mathsf{T}_{\mathsf{PLAN}}]$) //

1: **for all** $i \in \mathcal{V}$ **do** // Compute the value of (P-RTP) assuming (i) only robot i exists and (ii) no attacks will happen.

2: $J^{\star}_{\{i\},\,t,\,0} \triangleq \max_{\substack{u_{i,t'} \in \mathcal{U}_{i,t'} \\ t'=t+1:\,t+\mathsf{T}_{\mathsf{PLAN}}}} J_{\{i\},\,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}$;

3: Find a subset $\mathcal{L}$ of α robots such that $J^{\star}_{\{i\},\,t,\,0} \geq J^{\star}_{\{j\},\,t,\,0}$ for all $i \in \mathcal{L}$ and $j \in \mathcal{V} \setminus \mathcal{L}$; // $\mathcal{L}$ is the *bait* robot set ($\mathcal{L} \subseteq \mathcal{V}$ and $|\mathcal{L}| = \alpha$).

4: **for all** $i \in \mathcal{L}$ **do** // Assign to each robot $i \in \mathcal{L}$ the trajectory that achieves $J^{\star}_{\{i\},\,t,\,0}$.

5: $u_{i,\,t+1:t+\mathsf{T}_{\mathsf{PLAN}}} = \arg \max_{\substack{u_{i,t'} \in \mathcal{U}_{i,t'} \\ t'=t+1:\,t+\mathsf{T}_{\mathsf{PLAN}}}} J_{\{i\},\,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}$;

// Step 2: Remaining robots, $\mathcal{V} \setminus \mathcal{L}$, plan assuming (i) only robots in $\mathcal{V} \setminus \mathcal{L}$ exists and (ii) no attacks will happen //

6: $u_{\mathcal{V}\setminus\mathcal{L},\,t+1:t+\mathsf{T}_{\mathsf{PLAN}}} = \arg \max_{\substack{u_{\mathcal{V}\setminus\mathcal{L},t'} \in \mathcal{U}_{\mathcal{V}\setminus\mathcal{L},t'} \\ t'=t+1:\,t+\mathsf{T}_{\mathsf{PLAN}}}} J_{\mathcal{V}\setminus\mathcal{L},\,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}$.

assumes the robots in $\mathcal{L}$ are non-existent, and plans the control inputs for the remaining robots (RTP 's line 6).

E Performance Guarantees of RTP

Performance guarantees are unknown for RAIN , and, correspondingly, (P-RAIN) ((P-RAIN) is a mixed-integer, sequential control optimization problem, with limited a priori information on the measurements and attacks that are going to occur during the task-length, look-ahead time horizon). Nevertheless, in this section we quantify RTP 's performance, which is used by RAIN in a receding horizon fashion to approximate a solution to (P-RAIN) locally (cf. RAIN 's lines 3-15), by picking sequentially in time control inputs given (i) a shorter,

computationally feasible look-ahead time horizon (cf. Section C), and (ii) the history of the so far observed measurements and attacks.

Particularly, in this section we bound RTP 's approximation performance and running time. We use properties of the objective function $J_{\mathcal{V}\setminus\mathcal{A},\, t+1:t+\mathsf{T}_{\mathsf{PLAN}}}$ in (P-RTP) as a function of the set of robots; namely, the following notions of curvature.

E.1 Curvature and Total Curvature

We present the notions of curvature and total curvature for set functions. We start with the notions of modularity, and of non-decreasing and submodular set functions.

Definition 1 (Modularity [91]) Consider a finite (discrete) set $\mathcal{V}$. A set function $h : 2^{\mathcal{V}} \mapsto \mathbb{R}$ is modular if and only if $h(\mathcal{A}) = \sum_{v\in\mathcal{A}} h(v)$, for any $\mathcal{A} \subseteq \mathcal{V}$.

Hence, if h is modular, then $\mathcal{V}$'s elements complement each other through h. Specifically, Definition 1 implies $h(\{v\}\cup\mathcal{A}) - f(\mathcal{A}) = h(v)$, for any $\mathcal{A} \subseteq \mathcal{V}$ and $v \in \mathcal{V}\setminus\mathcal{A}$.

Definition 2 (Non-decreasing set function [91]) Consider a finite (discrete) set $\mathcal{V}$. $h : 2^{\mathcal{V}} \mapsto \mathbb{R}$ is non-decreasing if $h(\mathcal{B}) \geq h(\mathcal{A})$ for all $\mathcal{A} \subseteq \mathcal{B}$.

Definition 3 (Submodularity [91, Proposition 2.1]) Consider a finite (discrete) set $\mathcal{V}$. $h : 2^{\mathcal{V}} \mapsto \mathbb{R}$ is submodular if $h(\mathcal{A}\cup\{v\}) - h(\mathcal{A}) \geq h(\mathcal{B}\cup\{v\}) - h(\mathcal{B})$ for all $\mathcal{A} \subseteq \mathcal{B}$ and $v \in \mathcal{V}$.

Therefore, h is submodular if and only if the return $h(\mathcal{A}\cup\{v\}) - h(\mathcal{A})$ diminishes as $\mathcal{A}$ grows, for any v. If h is submodular, then $\mathcal{V}$'s elements substitute each other, in contrast to h being modular. Particularly, consider h to be non-negative (without loss of generality): then, Definition 3 implies $h(\{v\}\cup\mathcal{A}) - h(\mathcal{A}) \leq f(v)$. Thereby, v's contribution to $f(\{v\}\cup\mathcal{A})$'s value is diminished in the presence of $\mathcal{A}$.

Definition 4 (Curvature [87]) Consider a finite (discrete) $\mathcal{V}$, and a non-decreasing submodular $h : 2^{\mathcal{V}} \mapsto \mathbb{R}$ such that $h(v) \neq 0$ for any $v \in \mathcal{V}$, without loss of generality. Then, h's curvature

is defined as

$$\kappa_h \triangleq 1 - \min_{v \in \mathcal{V}} \frac{h(\mathcal{V}) - h(\mathcal{V} \setminus \{v\})}{h(v)}. \tag{4.6}$$

Definition 4 implies $\kappa_h \in [0,1]$. If $\kappa_h = 0$, then $h(\mathcal{V}) - h(\mathcal{V} \setminus \{v\}) = h(v)$, for all $v \in \mathcal{V}$, i.e., h is modular. Instead, if $\kappa_h = 1$, then there exist $v \in \mathcal{V}$ such that $h(\mathcal{V}) = h(\mathcal{V} \setminus \{v\})$, that is, v has no contribution to $h(\mathcal{V})$ in the presence of $\mathcal{V} \setminus \{v\}$. Overall, κ_h represents a measure of how much $\mathcal{V}$'s elements complement (and substitute) each other.

Definition 5 (Total curvature [88, Section 8]) Consider a finite (discrete) set $\mathcal{V}$ and a monotone $h : 2^{\mathcal{V}} \mapsto \mathbb{R}$. Then, h's total curvature is quantified as

$$c_h \triangleq 1 - \min_{v \in \mathcal{V}} \min_{\mathcal{A}, \mathcal{B} \subseteq \mathcal{V} \setminus \{v\}} \frac{h(\{v\} \cup \mathcal{A}) - h(\mathcal{A})}{h(\{v\} \cup \mathcal{B}) - h(\mathcal{B})}. \tag{4.7}$$

Definition 5 implies $c_h \in [0,1]$, similarly to Definition 5 for κ_h. When h is submodular, then $c_h = \kappa_h$. Generally, if $c_f = 0$, then h is modular, while if $c_h = 1$, then eq. (4.7) implies Definition 5's assumption that h is non-decreasing.

E.2 Performance Analysis for RTP

We quantify (i) suboptimality bounds on RTP 's approximation performance, and (ii) upper bounds on the running time RTP requires. We use the notation:

- $J^{\star}_{\mathcal{V},\, t,\, \alpha}$ is the optimal value of (P-RTP):

$$J^{\star}_{\mathcal{V},\, t,\, \alpha} \triangleq \max_{\substack{u_{\mathcal{V},t'} \in \mathcal{U}_{\mathcal{V},t'} \\ t' = t+1 \,:\, t+\mathsf{T}_{\mathsf{PLAN}}}} \min_{\mathcal{A} \subseteq \mathcal{V},\, |\mathcal{A}| \,\leq\, \alpha} J_{\mathcal{V} \setminus \mathcal{A},\, t+1:t+\mathsf{T}_{\mathsf{PLAN}}};$$

- $\mathcal{A}^{\star}$ is an optimal removal of α robots from $\mathcal{V}$ per (P-RTP):

$$\mathcal{A}^{\star} \triangleq \arg \min_{\mathcal{A} \subseteq \mathcal{V},\, |\mathcal{A}| \leq \alpha} J_{\mathcal{V} \setminus \mathcal{A},\, t+1:t+\mathsf{T}_{\mathsf{PLAN}}}.$$

We also use the definitions:

Definition 6 (Normalized set function [91]) Consider a discrete set $\mathcal{V}$. $h : 2^{\mathcal{V}} \mapsto \mathbb{R}$ is normalized if $h(\emptyset) = 0$.

Definition 7 (Non-negativeness [91]) Consider a discrete set $\mathcal{V}$. $h : 2^{\mathcal{V}} \mapsto \mathbb{R}$ is non-negative if $h(\mathcal{A}) \geq 0$ for all $\mathcal{A}$.

Theorem 1 (Performance of RTP) Consider an instance of (P-RTP). Assume the robots in $\mathcal{V}$ can solve optimally the (attack-free) information acquisition problem in eq. (4.5).

- Approximation performance: RTP returns control inputs $u_{\mathcal{V},\,1:t+\mathsf{T}_{\mathsf{PLAN}}}$ such that (i) if $J_{\cdot,t+1:t+\mathsf{T}_{\mathsf{PLAN}}} : 2^{\mathcal{V}} \mapsto \mathbb{R}$ is non-decreasing, and, without loss of generality, normalized and non-negative, then

$$\frac{J_{\mathcal{V}\setminus\mathcal{A}^\star,\,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}}{J^\star_{\mathcal{V},\,t,\,\alpha}} \geq (1 - c_{J_{\cdot,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}})^2; \tag{4.8}$$

  (ii) If, in addition, $J_{\cdot,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}$ is submodular, then

$$\frac{J_{\mathcal{V}\setminus\mathcal{A}^\star,\,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}}{J^\star_{\mathcal{V},\,t,\,\alpha}} \geq \max\left(1 - \kappa_{J_{\cdot,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}}, \frac{1}{1+\alpha}\right). \tag{4.9}$$

- Running time: If ρ upper bounds the running time for solving the (attack-free) information acquisition problem in eq. (4.5), then RTP terminates in $O(|\mathcal{V}|\rho)$ time.

Theorem 1's bounds in eqs. (4.8)-(4.9) compare RTP 's selection $u_{\mathcal{V},\,1:t+\mathsf{T}_{\mathsf{PLAN}}}$ against an optimal selection of control inputs that achieves the optimal value $J^\star_{\mathcal{V},\,t,\,\alpha}$ for (P-RTP). Particularly, eqs. (4.8)-(4.9) imply that for (i) non-decreasing and (ii) non-decreasing and submodular functions $J_{\cdot,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}$, RTP guarantees a value for (P-RTP) which can be close to the optimal. For example, eq. (4.9)'s lower bound $1/(1+\alpha)$ is non-zero for any finite number of robots $|\mathcal{V}|$, and, notably, it equals 1 in the attack-free case (RTP is exact for $\alpha = 0$, per

Theorem 1's assumptions). More broadly, when $\kappa_{J_{\cdot,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}} < 1$ or $c_{J_{\cdot,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}} < 1$, RTP 's selection $u_{\mathcal{V},\,1:t+\mathsf{T}_{\mathsf{PLAN}}}$ is close to the optimal, in the sense that Theorem 1's bounds are non-zero. Functions with $\kappa_{J_{\cdot,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}} < 1$ include the $\log\det$ of positive-definite matrices [92]; objective functions of this form are the conditional entropy and mutual information when used for batch-state estimation of stochastic processes [93]. Functions with $c_{J_{\cdot,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}} < 1$ include the average minimum square error (mean of the trace of a Kalman filter's error covariance across a finite time horizon) [94].

Theorem 1's curvature-dependent bounds in eqs. (4.8)-(4.9) also make a first step towards separating the classes of (i) non-decreasing and (ii) non-decreasing and submodular functions into functions for which (P-RTP) can be approximated well, and functions for which it cannot. Indeed, when either $\kappa_{J_{\cdot,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}}$ or $c_{J_{\cdot,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}}$ tend to zero, RTP becomes exact. For example, eq. (4.8)'s term $1-c_{J_{\cdot,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}}$ increases as $c_{J_{\cdot,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}}$ decreases, and its limit is equal to 1 for $c_{J_{\cdot,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}} \to 0$. Notably, however, the tightness of Theorem 1's bounds is an open problem. For example, although for the attack-free problem in eq. (4.5) a bound $O(1-c_{J_{\cdot,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}})$ is known to be optimal (the tightest possible in polynomial time and for a worst-case $J_{\cdot,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}$) [88, Theorem 8.6], the optimality of eq. (4.8) is an open problem.

Overall, Theorem 1 quantifies RTP 's approximation performance when the robots in $\mathcal{V}$ solve optimally the (attack-free) information acquisition problems in RTP 's line 2, line 5, and line 6. Among those, however, the problems in line 5 and line 6 are computationally challenging, being multi-robot coordination problems; only approximation algorithms are known for their solution. Such an approximation algorithm is the recently proposed coordinate descent [4, Section IV]. Coordinate descent has the advantages of having a provably near-optimal approximation performance. Therefore, we next quantify RTP 's performance when the robots in $\mathcal{V}$ solve the problems in RTP 's line 5, and line 6 using coordinate descent.[‡]

[‡]We refer to B for a description of coordinate descent.

Proposition 1 (Approximation Performance of RTP via Coordinate Descent) Consider an instance of (P-RTP). Assume the robots in $\mathcal{V}$ solve the (attack-free) information acquisition problem in eq. (4.5) suboptimally in the case of multiple robots ($|\mathcal{V}| \geq 2$) via coordinate descent [4, Section IV], and opti- mally in the case of a single robot ($|\mathcal{V}| = 1$). Then:

- Approximation performance: RTP returns control inputs $u_{\mathcal{V},\,1:t+\mathsf{T}_{\mathsf{PLAN}}}$ such that (i) if $J_{\cdot,t+1:t+\mathsf{T}_{\mathsf{PLAN}}} : 2^{\mathcal{V}} \mapsto \mathbb{R}$ is non-decreasing, and, without loss of generality, normalized and non-negative, then

$$\frac{J_{\mathcal{V}\setminus\mathcal{A}^\star,\,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}}{J^\star_{\mathcal{V},\,t,\,\alpha}} \geq \frac{1}{2}(1 - c_{J_{\cdot,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}})^3; \tag{4.10}$$

  (ii) If $J_{\cdot,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}$ is also submodular, then

$$\frac{J_{\mathcal{V}\setminus\mathcal{A}^\star,\,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}}{J^\star_{\mathcal{V},\,t,\,\alpha}} \geq \frac{1}{2}\max\left(1 - \kappa_{J_{\cdot,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}}, \frac{1}{1+\alpha}\right). \tag{4.11}$$

- Running time: If ρ_{CD} upper bounds the running time for solving the information acquisition problem in eq. (4.5) via coordinate descent, then RTP terminates in $O(\rho_{\mathsf{CD}})$ time.

Proposition 1's suboptimality bounds are discounted versions of Theorem 1's bounds: (i) eq. (4.10) is the discounted eq. (4.8) by the factor $(1 - c_{J_{\cdot,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}})/2$; and (ii) eq. (4.11) is the discounted eq. (4.9) by the factor $1/2$. The source of the discounting factors is the requirement in Proposition 1 that the robots in $\mathcal{V}$ can solve only suboptimally (via coordinate descent) the information acquisition problem in eq. (4.5) (and, in effect, the problems in RTP 's line 5 and line 6). In more detail, in Lemma 5, located in B, we prove that (i) for non-decreasing objective functions, coordinate descent guarantees the suboptimality bound $(1 - c_{J_{\cdot,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}})/2$ for eq. (4.5) (which is the discounting factor to eq. (4.8), resulting in eq. (4.10)), while (ii) for non-decreasing and submodular functions,

coordinate descent is known to guarantee the suboptimality bound 1/2 for eq. (4.5) (which is the discounting factor to eq. (4.9), resulting in eq. (4.11)) [4].

Proposition 1 also implies that if the robots in $\mathcal{V}$ use coordinate descent to solve the (attack-free) information acquisition problems in RTP 's line 5 and line 6, then RTP has the same of order of running time as coordinate descent. The proof of Proposition 1 is found in D.

F Applications and Experiments

We present RAIN 's performance in applications. We present three applications of Resilient Active Information Acquisition with Teams of Robots: (i) Multi-Target Tracking (Section F.1), (ii) Occupancy Grid Mapping (Section F.2), and (iii) Persistent surveillance (Section F.3). We confirm RAIN effectiveness, even as we vary key parameters in (P-RAIN):

(i) the number of attacks α, among the permissible values $\{0, 1, \ldots, |\mathcal{V}|\}$, to test RAIN 's performance to both small and high attack numbers (Section F.1);

(ii) the attack model, beyond the worst-case model prescribed by (P-RAIN)'s problem formulation, to test RAIN 's sensitivity against non worst-case failures; particularly, random failures (Section F.2).[§]

(iii) the replanning rate $\mathsf{T}_{\mathsf{REPLAN}}$, among the permissible values $\{1, 2, \ldots, \mathsf{T}_{\mathsf{PLAN}}\}$, to test RAIN 's performance even when the replanning rate is high (Section F.3).[¶]

Common Experimental Setup across Applications:

In the multi-robot case (pertained to RTP 's line 5, and line 6), the algorithm used for approximating a solution to eq. (4.5) is the coordinate descent [4] (also, cf. B). Evidently,

[§]For random failures, one would expect RAIN's performance to be the same, or improve, since RAIN is designed to withstand the worst-case.

[¶]When the replanning rate tends, ideally, to infinity, in that the robots can instantaneously observe all attacks and replan (which, however, is in practice impossible due to algorithmic, computational, and communication delays), then it is expected that RAIN's advantage over a non-resilient algorithm with the same replanning rate, such as coordinate descent, would diminish.

coordinate descent does not account for the possibility of attacks, and for this reason, we also use it as a baseline to compare RAIN with. In the single robot case (pertained to RTP 's line 2), eq. (4.5) reduces to a single-robot motion planning problem, and for its solution we use reduced value iteration (ARVI algorithm [1]), except for the application of Occupancy Grid Mapping (Section F.2) where we use forward value iteration [2]

Worst-Case Attack Approximation Computing the worst-case attack requires brute-force, since the minimization step in (P-RAIN) is NP-hard [95]. The consequence is that solving for the worst-case attack requires solving an exponential number of instances of the information acquisition problem in eq. (4.5), prohibiting real-time navigation performance by the robots, even for small teams of robots ($|\mathcal{V}| \geq 5$). In particular, the running time required to solve eq. (4.5), even via coordinate descent, can be exponential in the number of robots and task length horizon, namely, $\mathcal{O}(|\mathcal{U}|^{|\mathcal{V}|\mathsf{T}_{\mathsf{TASK}}})$ [4] ($\mathcal{U}$ denotes the set of admissible control inputs to each of the robots in $\mathcal{V}$, assumed the same across all robots). Hence, we approximate the worst-case attacks by solving the minimization step in (P-RAIN) via a greedy algorithm [21].

Computational Platform Experiments are implemented in C++, and run on an Intel Core i7 CPU laptop.

F.1 Resilient Multi-Target Tracking

In Resilient Multi-Target Tracking, a team of mobile robots is tasked to track the locations of multiple moving targets, even in the presence of a number of attacks against the robots. For the purpose of assessing RAIN 's effectiveness against various number of attacks, we will vary the number of attacks across scenarios, where we will also possibly vary the number of robots and targets. In more detail, the experimental setup and simulated scenarios are described below.

Experimental Setup. We specify the used (a) robot dynamics, (b) target process, (c) sensor model, and (d) information acquisition objective function:

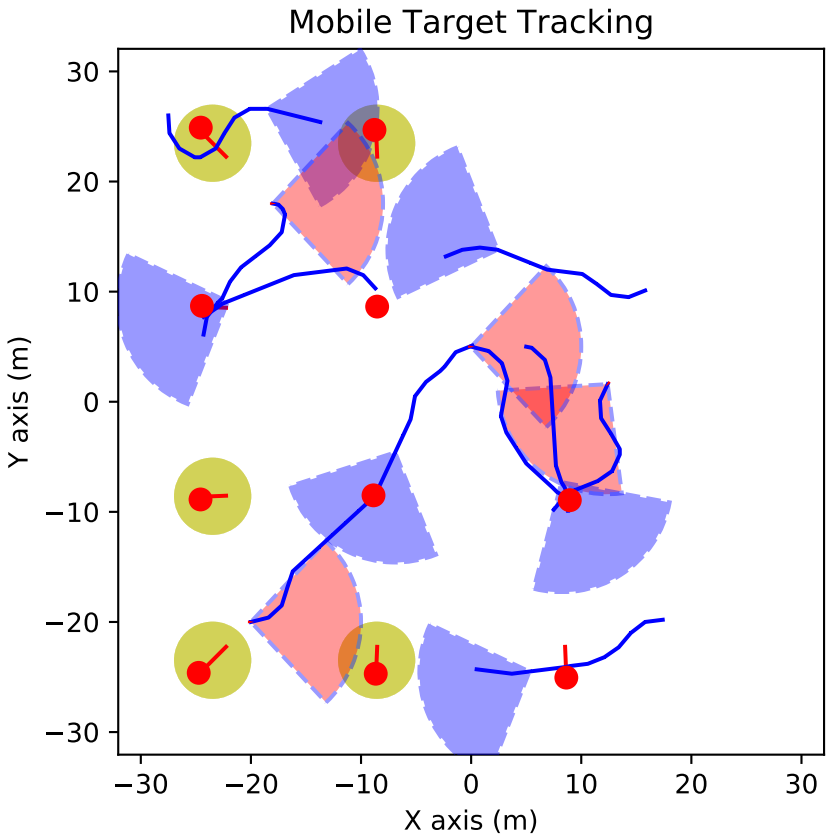


Figure 4.2. **Resilient Multi-Target Tracking Scenario**. 10 robots are depicted tracking 10 targets, while 4 of the robots are be attacked (causing their sensing capabilities to be, at least temporarily, disabled). The robots are depicted with their conic-shaped field-of-view, colored light blue for non-attacked robots and light red for attacked robots. The targets are depicted with red disks. Planned robot trajectories are shown as solid blue lines. Predicted target trajectories are shown as solid red lines. Each light-green ellipse represents the covariance of the target's location estimate.

Robot Dynamics Each robot i has unicycle dynamics in SE(2), discretized with a sampling period τ, such that

$$\begin{pmatrix} x_{i,t+1}^{(1)} \\ x_{i,t+1}^{(2)} \\ \theta_{i,t+1} \end{pmatrix} = \begin{pmatrix} x_{i,t}^{(1)} \\ x_{i,t}^{(2)} \\ \theta_{i,t} \end{pmatrix} + \begin{pmatrix} \nu_i \operatorname{sinc}(\frac{\omega_i \tau}{2}) \cos(\theta_{i,t} + \frac{\omega_i \tau}{2}) \\ \nu_i \operatorname{sinc}(\frac{\omega_i \tau}{2}) \sin(\theta_{i,t} + \frac{\omega_i \tau}{2}) \\ \tau \omega_i \end{pmatrix}, \tag{4.12}$$

where (ν_i, ω_i) is the control input (linear and angular velocity).

Target Dynamics The targets move according to double integrator dynamics, which are assumed corrupted with additive Gaussian noise. Specifically, if M denotes the number of targets, then $y_t = [y_{t,1}^\top, \ldots, y_{t,M}^\top]^\top$, where $y_{t,m}$ is the planar coordinates and velocities of target m, and

$$y_{t+1,m} = \begin{bmatrix} I_2 & \tau I_2 \\ 0 & I_2 \end{bmatrix} y_{t,m} + w_t, \quad w_t \sim \mathsf{N}\left(0, q \begin{bmatrix} \tau^3/3 I_2 & \tau^2/2 I_2 \\ \tau^2/2 I_2 & \tau I_2 \end{bmatrix}\right).$$

with q being a noise diffusion parameter.

Sensor Model The robots' sensor model consists of a range and bearing for each target $m = 1, \ldots, M$:

$$\begin{aligned} z_{t,m} &= h(x_t, y_{t,m}) + v_t, \quad v_t \sim \mathsf{N}(0, V(x_t, y_{t,m})); \\ h(x, y_m) &= \begin{bmatrix} r(x, y_m) \\ \alpha(x, y_m) \end{bmatrix} \triangleq \begin{bmatrix} \sqrt{(y^1 - x^1)^2 + (y^2 - x^2)^2} \\ \tan^{-1}((y^2 - x^2)(y^1 - x^1)) - \theta \end{bmatrix}. \end{aligned}$$

Since the sensor model is non-linear, we linearize it around the currently predicted target trajectory. Particularly, given

$$\nabla_y h(x, y_m) = \frac{1}{r(x, y_m)} \begin{bmatrix} (y^1 - x^1) & (y^2 - x^2) & 0_{1x2} \\ -\sin(\theta + \alpha(x, y_m)) & \cos(\theta + \alpha(x, y_m)) & 0_{1x2} \end{bmatrix},$$

the observation model for the joint target state can be expressed as a block diagonal matrix containing the linearized observation models for each target along the diagonal:

$$H \triangleq \text{diag}\nabla_{y_1} h(x, y_1), \ldots, \nabla_{y_M} h(x, y_M).$$

The sensor noise covariance grows linearly in range and in bearing, up to σ_r^2, and σ_b^2, where σ_r and σ_b are the standard deviation of the range and the bearing noise, respectively. The model here also includes a limited range and field of view, denoted by the parameters r_{sense} and ψ, respectively.

Information Acquisition Objective Function For the information objective function we use the time averaged log determinant of the covariance matrix, which is equivalent to conditional entropy for Gaussian variables[2]. This objective function is non-decreasing, yet not necessarily submodular [96]. Overall, we solve an instance of (P-RAIN) per the aforementioned setup and the objective function:

$$J_{\mathcal{V},\,1:\mathsf{T}_{\mathsf{TASK}}} \triangleq \frac{1}{\mathsf{T}_{\mathsf{TASK}}} \sum_{t=1}^{\mathsf{T}_{\mathsf{TASK}}} \log\det(\Sigma_{\mathcal{V},\,t}),$$

where $\Sigma_{\mathcal{V},\,t}$ is the Kalman filtering error covariance at t, given the measurements collected up to t by the robots in $\mathcal{V}$ [2].[‖]

Simulated Scenarios. We consider multiple scenarios of the experimental setup introduced above: across scenarios, we vary the number of robots, n, the number of targets M, and the number of attacks, α; cf. first column of Table 4.1. Additionally, the robots and targets are restricted to move inside a $64 \times 64m^2$ environment (Fig. 4.2). The admissible control input values to each robot are the $\mathcal{U} = \{1,3\}m/s \times \{0, \pm 1, \pm 3\}rad/s$. At the beginning of each scenario, we fix the initial positions of both the robots and targets, and the robots are

[‖]The multi-target tracking scenarios are dependent on a prior distribution of the target's initial conditions y_0 and $\Sigma_{0|0}$, assumed here known. Yet, if a prior distribution is unknown, then an exploration strategy can be incorporated to find the targets by placing exploration landmarks at the map frontiers [4].

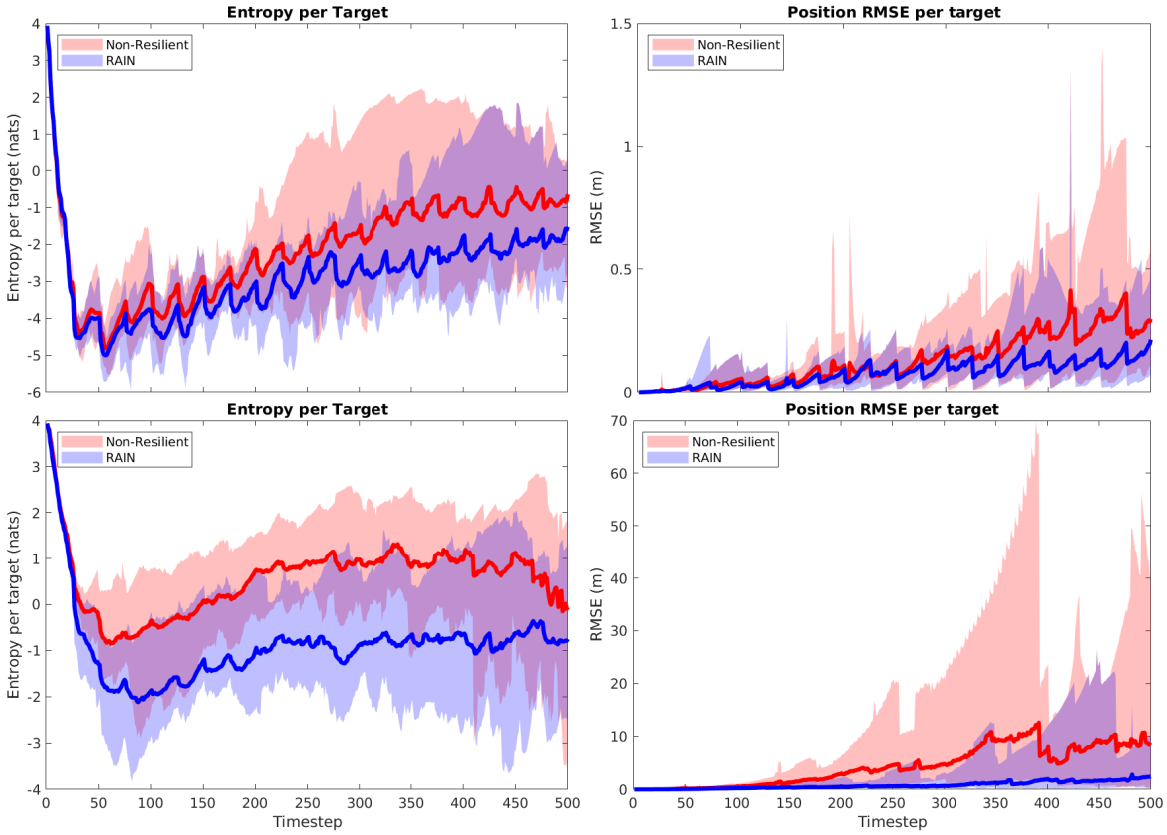


Figure 4.3. **Resilient Multi-Target Tracking Results.** Performance comparison of RAIN with coordinate descent (noted as NonResilient in the plots), for two configurations and two performance metrics: top row considers 10 robots, 10 targets, and 2 attacks; bottom row considers the same number of robots and targets but 6 attacks. The left column depicts mean entropy per target, averaged over the robots; the right column depicts the position Root Mean Square Error (RMSE) per target, also averaged over the robots.

given a prior distribution of the targets before starting the simulation. The targets start with a zero velocity, and in the event that a target leaves the environment its velocity is reflected to remain in bounds. Finally, across all simulations: $\mathsf{T}_{\mathsf{PLAN}} = \mathsf{T}_{\mathsf{REPLAN}} = 25$ steps, $\tau = 0.5s$, $r_{sense} = 10m$, $\psi = 94^\circ$, $\sigma_r = .15m$, $\sigma_b = 5^\circ$, and $q = .001$. We use $\mathsf{T}_{\mathsf{TASK}} = 500$.

Compared Techniques. We compare RAIN with coordinate descent. We consider two performance measures: the average entropy, and average Root Mean Square Error (RMSE) per target, averaged over the robots in the team.

	Mean RMSE		Peak RMSE	
	NonRes	RAIN	NonRes	RAIN
$n = 5, M = 10$				
$\alpha = 1$	0.28	0.19	9.62	2.09
$\alpha = 2$	1.47	0.68	26.07	15.71
$\alpha = 4$	10.67	4.9	225.47	103.82
$n = 10, M = 5$				
$\alpha = 2$	0.35	0.14	57.65	1.87
$\alpha = 4$	0.39	0.28	6.66	3.17
$\alpha = 6$	2.07	0.65	93.27	15.63
$n = 10, M = 10$				
$\alpha = 2$	0.13	0.08	1.4	1.32
$\alpha = 4$	0.24	0.23	4.19	2.66
$\alpha = 6$	4.39	1.2	69.77	26.4

Table 4.1: **Resilient Multi-Target Tracking Results.** Performance comparison of RAIN with coordinate descent (noted as NonRes in the table), for a variety of configurations, where n denotes the number of robots ($n = |\mathcal{V}|$), M denotes the number of targets, and α denotes the number of failures. Two performance metrics are used: mean Root Mean Square Error (RMSE), and peak RMSE, both per target, and averaged over the robots in the team.

Results. The results, averaged across 10 Monte-Carlo runs, are depicted in Fig. 4.3 and Table 4.1. In Fig. 4.3, RAIN 's performance is observed to be superior both in terms of the average entropy and the RMSE. Particularly, as the number of attacks grows (cf. second rows of plots in Fig. 4.3), RAIN 's benefits are accentuated in comparison to the non-resilient coordinate descent. Similarly, Table 4.1 demonstrates that RAIN achieves a lower mean RMSE than coordinate descent, and, crucially, is highly effective in reducing the peak estimation error; in particular, RAIN achieves a performance that is 2 to 30 times better in comparison to the performance achieved by the non-resilient algorithm. We also observe that the impact of Algorithm 10 is most prominent when the number of attacks is large relative to the size of the robot team.

F.2 Resilient Occupancy Grid Mapping

We show how (P-RAIN)'s framework for resilient active information acquisition can be adapted to exploring an environment when the map and objective are defined via occupancy

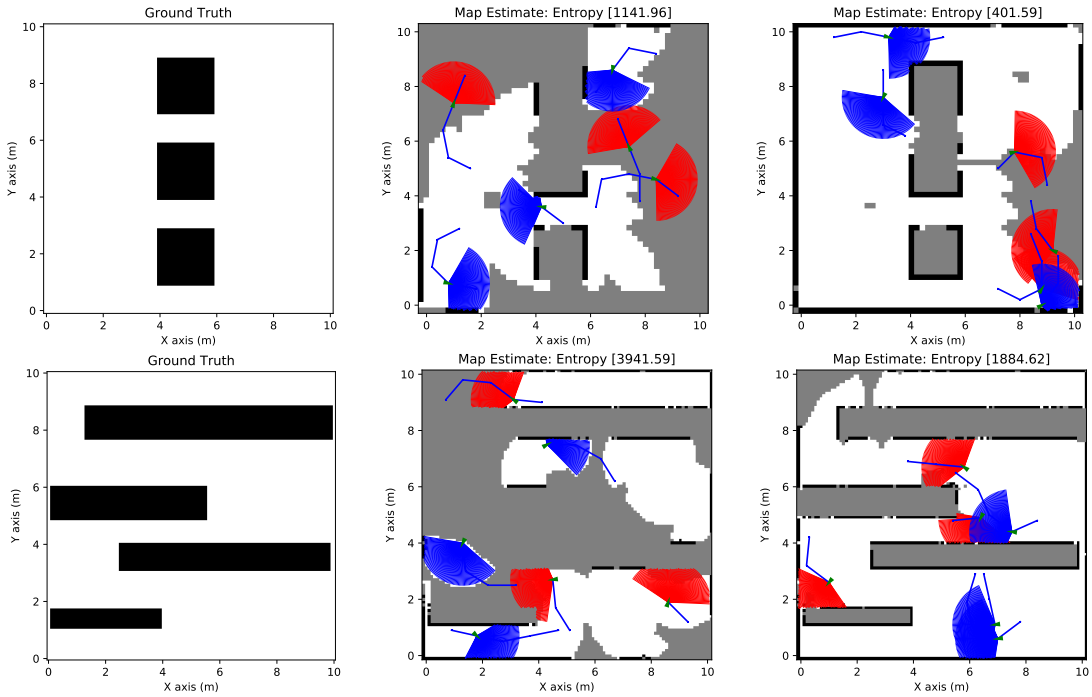


Figure 4.4. **Resilient Occupancy Grid Mapping Scenarios**. Two scenarios are considered: a square obstacle map (top row), and a corridor map (bottom row), where free space is colored white, occupied space is colored black, and unexplored/unknown space is colored gray. The non-attacked robots are shown with their field-of-view colored blue, whereas the attacked robots are shown with field-of-view colored red. The left-most column shows the considered ground truth maps; the middle column shows the map estimate halfway through the task horizon; the right-most column shows the map estimate near completion.

grids. In this section, we also assess RAIN 's sensitivity against non worst-case attacks, in particular, random.

Experimental Setup. We specify the used (a) robot dynamics, (b) target process, (c) sensor model, (d) information acquisition objective function, and (e) algorithm for solving the optimization problem in RTP 's line 2:

Robot Dynamics The robots' dynamics are as in the multi-target tracking application (Section F.1).

Target Process We define the target process y_t, which we will denote henceforth as M for consistency with common references on occupancy grid mapping [97], where we also drop the time subscript since the process y_t does not evolve in time. The occupancy grid M is a 2-

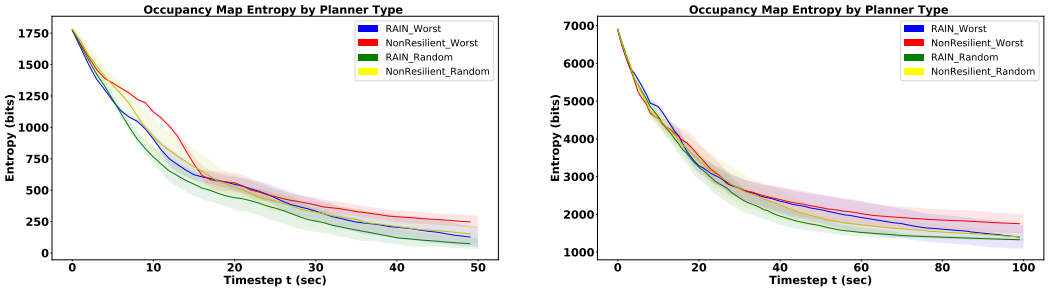


Figure 4.5. **Resilient Occupancy Mapping Results.** Comparison of achieved entropy by RAIN against coordinate descent (noted as NonResilient in the plots) for increasing time and for two types of attacks, worst-case and random: (left plot) result for the square obstacles map (top row of Fig. 4.4); (right plot) result for the corridor map (bottom row of Fig. 4.4).

dimensional grid with n rows and m columns, discretized into cells $M = \{C_1, \ldots, C_{n_m}\}$, which are binary variables that are either occupied or free, with some probability. Cell occupancy is assumed to be independent, so that the probability mass function can be factored as $\mathbb{P}(M = m) = \prod_{i=1}^{nm} \mathbb{P}(C_i = c_i)$, where $c_i \in \{0,1\}$, and where $m \in \{0,1\}^{nm}$ is a particular realization of the map.

Sensor Model We express the sensor model as a series of B beams, such that $z_t = [z_t^1, \ldots, z_t^B]$, where z_k^b is the random variable of the distance that a beam b travels to intersect an object in the environment. We next define the distribution for a single beam, determined by the true distance d to the first obstacle that the beam intersects:

$$p(z_k^b = z|d) = \begin{cases} \mathcal{N}(z-0, \sigma^2), & d < z_{\min}; \\ \mathcal{N}(z-z_{max}, \sigma^2), & d > z_{\max}; \\ \mathcal{N}(z-d, \sigma^2), & \text{otherwise}; \end{cases} \tag{4.13}$$

$z_{\min}$ and $z_{\max}$ are the minimum and maximum sensing ranges.

Information Acquisition Objective Function The used information objective function is the Cauchy Schwarz Quadratic Mutual Information (CSQMI), which is shown in the literature to be computationally efficient, as well as, sufficiently accurate for occupancy mapping [16].

We denote the CSQMI gathered at time t by $I_{CS}(m; z_{\mathcal{V},t})$, given the measurements collected at t by the robots in $\mathcal{V}$. Then,

$$J_{\mathcal{V},1:\mathsf{T}_{\mathsf{TASK}}} \triangleq \frac{1}{\mathsf{T}_{\mathsf{TASK}}} \sum_{t=1}^{\mathsf{T}_{\mathsf{TASK}}} I_{CS}(m; z_{\mathcal{V},t}),$$

.

Remark 16 (Evaluation of CSQMI) Details on the evaluation of the CSQMI objective are beyond the scope of this paper; we refer the reader to [16]. We note that it relies on a ray-tracing operation for each beam, computed over the current occupancy grid map belief to determine which cells each beam from the LIDAR will observe when the sensor visits a given pose. CSQMI is approximated by assuming the information computed over single beams is additive, but not before pruning approximately independent beams. This removal of approximately independent beams encourages robots to explore areas where their beams will not overlap. In coordinate descent, once a set of prior robots has planned trajectories, future robots must check their beams to see if they are approximately independent from the fixed beams, before their individual beam contributions may be added to the joint CSQMI objective.

Algorithm for Solving Optimization Problem in RTP 's line 2 The single-robot motion planning is performed via a full forward search over a short planning horizon $\mathsf{T}_{\mathsf{PLAN}} = 4$, since scaling beyond short horizons is challenging in occupancy mapping problems; for details, cf. [16]. We remark that the performance guarantees do not explicitly hold for the single-robot planner since the measurement model is highly non-linear and the cost function depends on the realization of the measurements, so open-loop planning is not optimal as it is with the Gaussian case [2]. Nonetheless, approaches similar to what we adopt here have been successfully used for (attack-free) occupancy grid mapping [3, 5].

Simulated Scenarios. To evaluate the performance of the resilient occupancy mapping

algorithm, we compare the results with different attack types. Namely, we consider an attack model where the attacks on robots may be uniformly random, rather than the worst case attack assumption of the previous section and the algorithm itself. In the experiment, the robots choose trajectories composed of $\mathsf{T}_{\mathsf{PLAN}} = 4$ steps of duration $\tau = 1$ sec, with motion primitives $\mathcal{U} = \{(v, \omega) : v \in \{0, 1, 2\} m/s,\ \omega \in \{0, \pm 1, \pm 2\} rad/s\}$. The maximum sensor range r_{sense} is $1.5m$, with a noise standard deviation of $\sigma = .01m$. The experiment considers a team of six robots, subject to two attack models, described in the following paragraph. The attacked set of robots is re-computed at the end of each planning duration. We evaluate the performance on two map environments, which we will refer to as the square obstacles map (Fig 4.4, top), and the corridor map (Fig 4.4, bottom). We use $\mathsf{T}_{\mathsf{TASK}} = 50$ and $\mathsf{T}_{\mathsf{TASK}} = 100$ for the squares and corridor map respectively, and $\mathsf{T}_{\mathsf{REPLAN}} = \mathsf{T}_{\mathsf{TASK}}$.

Compared Attack Models. We test RAIN 's ability to be effective even against non worst-case failures. To this end, beyond considering the worst-case attack model prescribed by (P-RAIN)'s problem formulation (cf. red and blue curves in Fig. 4.5), we also consider random attacks, chosen with uniformly random assignment among the robots in $\mathcal{V}$, given the attacks number α (cf. green and yellow curves Fig. 4.5).

Results. The results, averaged across 50 Monte-Carlo runs, are shown in Fig. 4.5. The plots indicate RAIN always improves performance. Specifically, RAIN improves performance both when (i) worst-case attacks are present (cf. blue and red curves in Fig. 4.5), and when (ii) random attacks are present (cf. green and yellow curves in Fig. 4.5); in both cases, RAIN attains lesser map entropy against coordinate descent (noted as NonResilient in the plots). Both the square obstacles map (left plot in Fig. 4.5) and the corridor map (right plot in Fig. 4.5) support this conclusion. Moreover, Fig. 4.5 supports the intuition that since RAIN is designed to withstand the worst-case attacks, RAIN 's performance will improve when instead only random failures are present (cf. blue and green curves in Fig. 4.5).

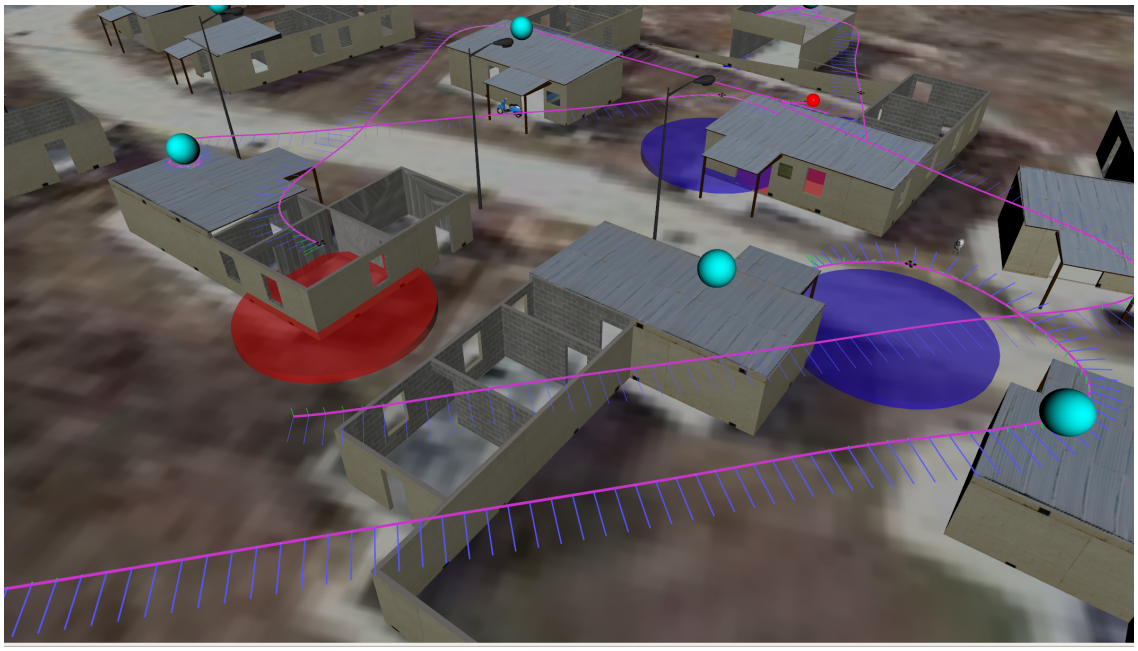


Figure 4.6. **Resilient Persistent Surveillance Scenario.** Camp Lejeune 3D environment. The robots' trajectories are shown in pink. The blue lines along the trajectories indicate the velocity profile generated along the trajectory. The blue discs show the field-of-view of the non-attacked robots. A red colored field-of-view indicates an attacked robot. The cyan spheres represent the relative uncertainty on the landmarks' locations. The landmarks are depicted as the red spheres.

F.3 Resilient Persistent Surveillance

In Resilient Persistent Surveillance, the robots' objective is to re-visit a series of static and known landmarks, while the robots are under attack. The landmarks represent points of interest in the environment. For example, a team of robots may be tasked to monitor the entrances to buildings for intruders [98]; the task becomes especially interesting as the number of entrances becomes more than the number of robots. In this section, we choose the landmarks to be a set of buildings in an outdoor Camp environment (Fig. 4.6). We use the simulated scenarios to determine the effect of the replanning rate on RAIN 's performance (cf. Footnote ¶).

Experimental Setup. The environment used is a 3D environment provided by ARL DCIST (Fig. 4.6). It contains a set of outdoor buildings, over which we place landmarks to

encourage visitation (one landmark per building). To have the robots (re-)visit the landmarks, we add artificial uncertainty to the location of each landmark by proportionally increasing the uncertainty with time passed since the last observation. The software simulation stack used is based on the Robot Operating System (ROS); the back-end physics are based on Unity. In all experiments, the map is assumed to be known. Localization is provided by the simulator. We next specify the used (a) robot dynamics, (b) target process, (c) sensor model, and (d) information acquisition objective function:

Robot Dynamics The robot motion model is adapted from the 2D in eq. (4.12) to the following 3D:

$$\begin{pmatrix} x^1_{t+1} \\ x^2_{t+1} \\ x^3_{t+1} \\ \theta_{t+1} \end{pmatrix} = \begin{pmatrix} x^1_t \\ x^2_t \\ x^3_t \\ \theta_t \end{pmatrix} + \begin{pmatrix} \nu\,\mathrm{sinc}(\frac{\omega\tau}{2})\cos(\theta_t + \frac{\omega\tau}{2}) \\ \nu\,\mathrm{sinc}(\frac{\omega\tau}{2})\sin(\theta_t + \frac{\omega\tau}{2}) \\ 0 \\ \tau\omega \end{pmatrix}.$$

That is, we assume the quadrotors to fly at a fixed height over the environment ($x^3_{t+1} = x^3_t$) always).

Target Process The targets are assumed static in location, but corrupted with uncertainty that increases over time to encourage (re-)visitation by the robots, according to a noise covariance matrix $qk_{t,m}I_3$, where q is the rate of uncertainty increase, and $k_{t,m}$ denotes the number of time steps since target m was last visited.

$$y_{t+1,m} = y_{t,m} + w_t, \quad w_t \sim \mathsf{N}(0, qk_{t,m}I_3).$$

Sensor Model We assume the robots operate a 360° field-of-view downward facing sensor. In particular, we assume a range sensing model that records information as long as the robots are within some radius r_{sense} from a landmark; otherwise, no information is granted

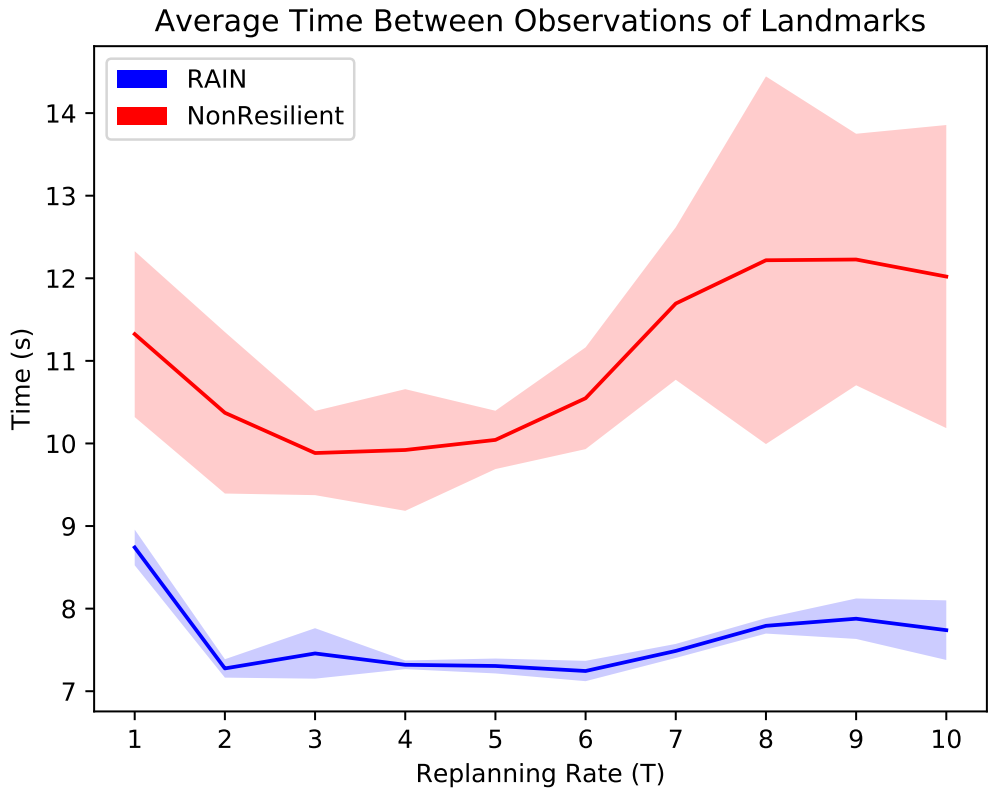


Figure 4.7. **Resilient Persistent Surveillance Results.** Comparison of average time between consecutive observations of the same landmarks by RAIN and coordinate descent (noted as NonResilient in the plot) for increasing replanning values $\mathsf{T}_{\mathsf{REPLAN}}$.

to the robot. The range based model for detecting the buildings is as follows:

$$z_{t,m} = h(x_t, y_{t,m}) + v_t, \quad v_t \sim \mathsf{N}(0, V(x_t, y_{t,m}));$$
$$h(x, y_m) = \left[r(x, y_m) \right] \triangleq \sqrt{\sum_{i=1}^{3} (y^1 - x^1)^2}.$$

Information Acquisition Objective Function We use the same information acquisition objective function as in the Multi-Target Tracking scenarios (Section F.1).

Simulation Setup. The admissible control inputs are the $\mathcal{U} = \{(v, \omega) : v \in \{1,3\} m/s, \omega \in \{0,1,2\}\}$. r_{sense} is 10 meters, the task duration is $\mathsf{T}_{\mathsf{TASK}} = 300$ steps, and the planning

horizon is $\mathsf{T}_{\mathsf{PLAN}} = 10$ steps. Specifically, each timestep has duration $\tau = 1s$. The noise parameter is $q = .01$.

Performance Metric. We measure RAIN 's performance by computing the average number of timesteps that a building goes unobserved for. For example, if a landmark is observed at timestep k, and not observed again until timestep $k+l$, we record l as the number of timesteps the landmark was unobserved. Particularly, we average these durations across all targets and timesteps for a given experimental trial.

Results. The results, averaged across 50 Monte-Carlo runs, are shown in Fig. 4.7. In Fig. 4.7, we observe even for the highest replanning rate ($\mathsf{T}_{\mathsf{REPLAN}} = 1$), RAIN offers a performance gain of $\simeq 24\%$ in comparison to coordinate descent (noted as NonResilient in Fig. 4.7). The gain increases on average, the lower the replanning rate becomes, as expected (cf. Footnote ¶). More broadly, Fig. 4.7 supports the intuition that a higher replanning rate allows even a non-resilient algorithm, such as coordinate descent, to respond to attacks rapidly, and thus perform well. Still, in Fig. 4.7 RAIN dominates coordinate descent across all possible replanning rate values.

F.4 Experiments on multi-target tracking with mobile robots

We implement Algorithm 10 in a multi-UAV scenario with two quadrotors tracking the positions of two static ground targets, shown in Fig. 4.8. The UAV trajectories are computed off-board but in real-time on a laptop with an Intel Core i7 CPU. The UAVs are localized using the Vicon Motion Capture system. The UAVs are quad-rotors equipped with Qualcomm Flight™. The UAVs use Vicon pose estimates to generate noisy measurements corresponding to a downward facing camera which has a 360° field-of-view, and a 1 meter sensing radius. The UAVs move in a 4x8 meter testing laboratory environment with no obstacles. One robot is jammed at all times.

The goal of the hardware experiments is to acquire a visual interpretation of the properties of the trajectories designed using the resilient Algorithm 10. To isolate the effect of

resilience, we simplify the problem to static targets (i.e. stationary) and to the smallest possible team, i.e., 2 robots.

We observe from the experiments that the trajectories planned by the UAVs under the non-resilient algorithm stick to the target they are closest to, whereas under the resilient Algorithm 10, the UAVs switch amongst the two targets (Fig. 4.9). Intuitively, the reason is that the resilient algorithm always assumes that one of the robots will fail, in which case the optimal strategy for one UAV is to track two targets is to switch amongst the targets, whereas the non-resilient algorithm assumes that none of the robots will fail, in which case the optimal strategy for two UAVs is to allocate themselves to the closest target. When there is the possibility of one UAV failing, switching amongst the targets is preferable, since both robots will acquire information about both targets.

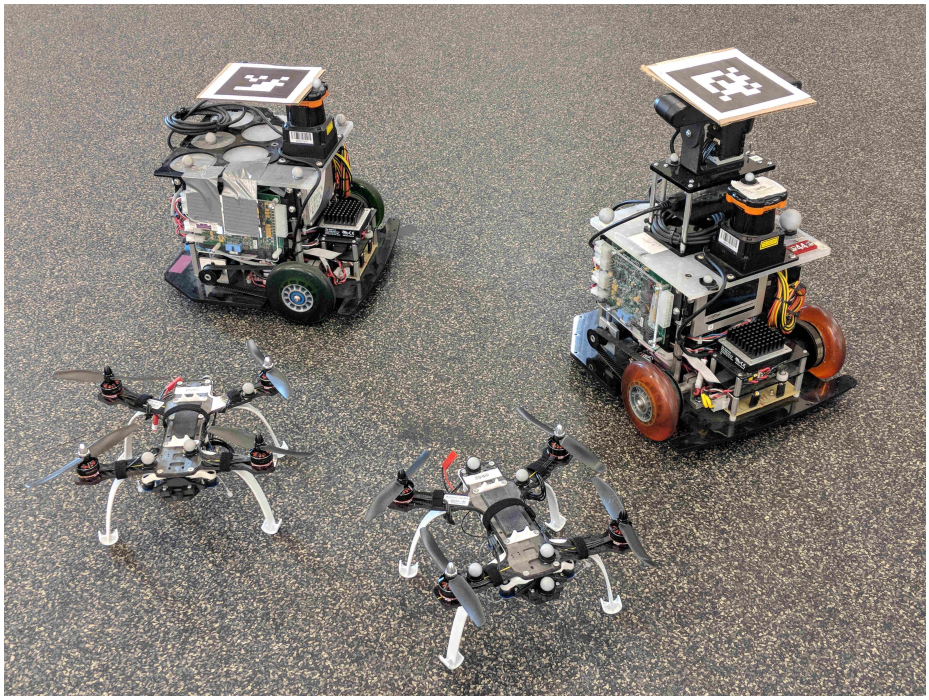
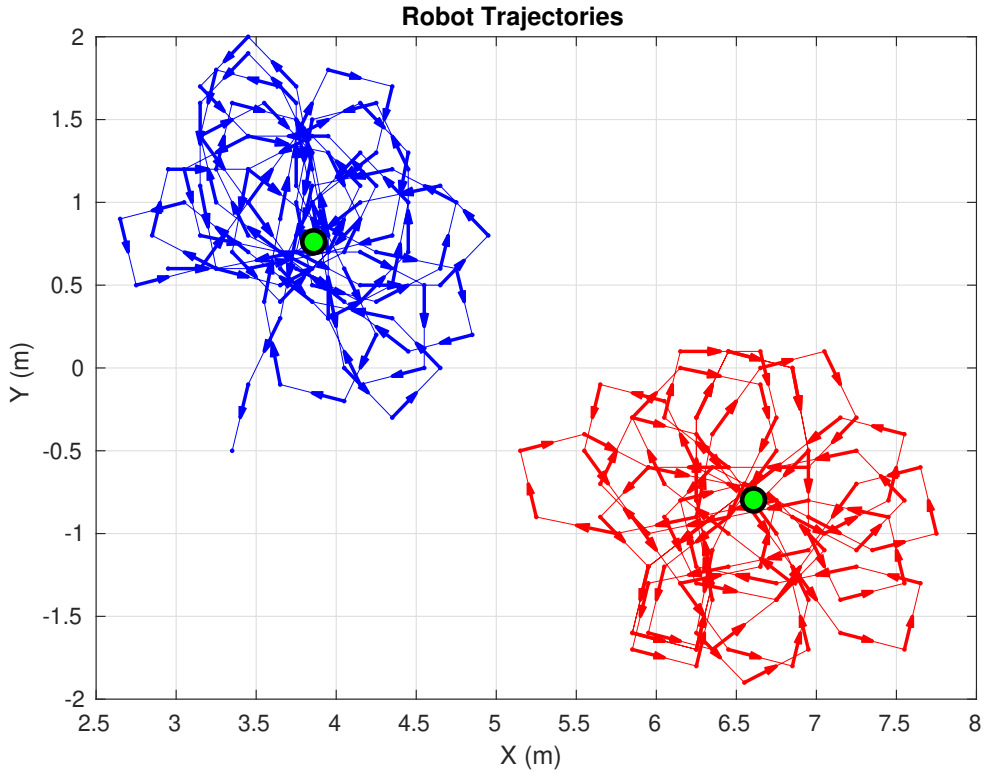
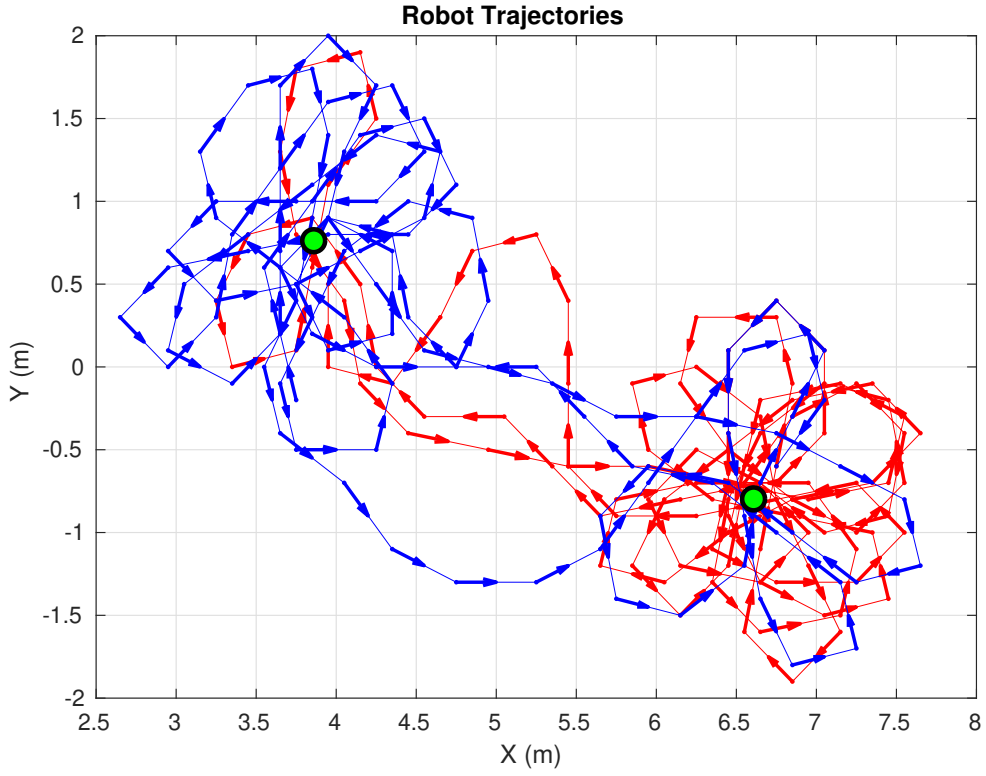


Figure 4.8. The experimental setup with two quad-rotors equipped with Qualcomm Flight™, and two Scarabs as ground targets.





(b)

Figure 4.9. The plot in (a) depicts the experimental robot trajectories in the non-resilient algorithm. The figure in (b) depicts the resilient algorithm. The targets are in green.

G Conclusion

We introduced the first receding-horizon framework for resilient multi-robot path planning against attacks that disable robots' sensors during information acquisition tasks (cf. (P-RAIN)). We proposed Resilient Active Information acquisitioN (RAIN), a robust and adaptive multi-robot planner against any number of attacks. RAIN calls, in an online fashion, Robust Trajectory Planning (RTP), a subroutine that plans attack-robust control inputs over a look-ahead planning horizon. We quantified RTP 's performance by bounding its suboptimality, using notions of curvature for set function optimization. We demonstrated the necessity for resilient multi-robot path planning, as well as RAIN 's effectiveness, in information acquisition scenarios of Multi-Target Tracking, Occupancy Grid Mapping, and Persistent Surveillance. In all simulations, RAIN was observed to run in real-time, and exhibited superior performance against a state-of-the-art baseline, (non-resilient) coordinate descent [4]. Across the three scenarios, RAIN 's exhibited robustness and superiority even (i) in the presence of a high number of attacks, (ii) against varying models of attacks, and (iii) high replanning rates. Future work includes extending the proposed framework and algorithms to distributed settings [5, 26].

Chapter 5

Conclusions and Future Work

In this dissertation, we propose a set of algorithmic tools that efficiently solve resilient, multi-robot, and non-myopic information acquisition problems. The tools in each chapter build on each other, and when put together can solve interesting and important information acquisition problems across a breadth of domains. We leverage mathematical tools such as submodularity, monotonicity, and concavity, as well as properties of classical robotics algorithms such as A* with consistent and admissible heuristic functions designed for information gathering problems.

Example applications of this thesis include localization and mapping, target tracking, environmental monitoring, search and rescue, and surveillance, several of which have been implemented here in laboratory simulations and hardware experiments. The approaches can be made fully distributed, in both the planning and control aspects, further adding to the practicality of the algorithms.

This thesis does have limitations in the form of some common assumptions throughout, namely the requirement of known robot dynamics, target dynamics, and sensor observation models. Real world models are complex, and increasingly being learned from data [99]. Furthermore, we typically consider target models that can be captured accurately with Gaussian distributions, with the exception of the Occupancy grid work in Chapter

4. Additionally, this line of work focuses exclusively on search-based algorithms, although there is a new family of algorithms that are based on sampling [23, 24]. It is not clear at this point, which family of algorithms performs better in different scenarios.

Future work may push further on the assumptions of known dynamics and observation models, as well as the limitations of Gaussian target belief distributions. Additionally, more work is needed in order to determine which types of problems the search-based or sampling based information acquisition algorithms are most well suited for.

Chapter 6

Appendix and Proofs

The following definitions and lemmas will be necessary for proving Theorem 3:

Definition 17 (ε,δ)-redundancy) A node (x, Σ) is (ε,δ)-redundant with respect to a set $\{(x_i,\Sigma_i)\}$ if Σ is ε-algebraically redundant with respect to all Σ_i whose corresponding x_i, δ-cross x.

Lemma 18 After each timestep t in ImprovePath(ε,δ), the set O_t will contain all non-(ε,δ)-redundant nodes from S_t.

Proof: The loop in line 10 of Alg. 2 evaluates all states (x, Σ) that are in S_t, but not in O_t, and checks (ε,δ)-redundancy with respect to O_t. If a node is non-(ε,δ)-redundant, it is added to O_t in line 14 of Alg. 2.

Lemma 19 After each timestep t, C_{t-1} indicates the nodes that have been expanded, and all nodes contained in O_{t-1} will also be contained in C_{t-1}. S_t contains the successors of each node in O_{t-1}.

Proof: The loop in line 2 of Alg. 2 evaluates all nodes (x, Σ) that are in O_{t-1} but not in C_{t-1}, and adds them to C_{t-1}, in addition to evaluating the motion model along the nodes and placing the result in S_t.

Proof of Theorem 3

To prove part (i) of Thm. 3, we note that after line 4 of Alg. 1, all sets are empty except S_0 and O_0, which both contain the initial node (x_0, Σ_0). No elements are ever removed from a set, and sets grow as ImprovePath is called with decreasing pruning parameters.

In a call to ImprovePath, the set O_t contains all non-(ε,δ)-redundant nodes (Lemma 1), and C_t contains all nodes previously searched (Lemma 2). Thus the set $O_t \setminus C_t$ contains all not previously searched non-(ε,δ)-redundant nodes, which guarantees an (ε,δ)-optimal solution will be produced.

To prove the monotonicity property (ii), we recall the (ε, δ)-suboptimality bound ([15]):

$$\begin{aligned} 0 \leq J_T^{\varepsilon,\delta} - J_T^* \leq (\zeta_T - 1)\left[J_T^* - \log\det(W)\right] + \varepsilon\Delta_T \\ \text{where} \quad \zeta_T := \prod_{\tau=1}^{t-1}(1 + \sum_{s=1}^{\tau} L_f^s L_m \delta) \geq 1 \end{aligned} \tag{6.1}$$

and Δ_T, L_f^s and L_m are constants. The suboptimality gap (6.1) is monotone in (ε,δ). This implies that the cost itself, $J_T^{\varepsilon,\delta}$ decreases monotonically in (ε,δ). In line 9 of Alg. 2, (ε,δ) decrease monotonically in runtime, which implies that the solution cost decreases monotonically in runtime.

Finally, to prove (iii) we note that in Main, $(\varepsilon,\delta) \to (0, 0)$, which is known to preserve optimality ([15]). From the first part of the the theorem, it is guaranteed that any call to ImprovePath will return an (ε,δ) optimal solution. Thus, given enough time ImprovePath(0,0) will be called and return the optimal solution.

In the appendices that follow, we prove Theorem 1 (C) and Proposition 1 (D). To this end, we first present supporting lemmas (A) and the algorithm coordinate descent [4] (B). We also use the notation:

Notation. Consider a finite set $\mathcal{V}$ and a set function $f : 2^{\mathcal{V}} \mapsto \mathbb{R}$. Then, for any set $\mathcal{X} \subseteq \mathcal{V}$ and any set $\mathcal{X}' \subseteq \mathcal{V}$, the symbol $f(\mathcal{X}|\mathcal{X}')$ denotes the marginal value $f(\mathcal{X} \cup \mathcal{X}') - f(\mathcal{X}')$.

We also introduce notation emphasizing that subsets of robots may use different algorithms to compute their

A Preliminary Lemmas

The proof of the lemmas is also found in [85, 100].

Lemma 1 Consider a finite set $\mathcal{V}$ and a non-decreasing and submodular set function $f : 2^{\mathcal{V}} \mapsto \mathbb{R}$ such that f is non-negative and $f(\emptyset) = 0$. For any $\mathcal{A} \subseteq \mathcal{V}$:

$$f(\mathcal{A}) \geq (1 - \kappa_f) \sum_{a \in \mathcal{A}} f(a).$$

*Proof of Lemma 1 Let $\mathcal{A} = \{a_1, a_2, \ldots, a_{|\mathcal{A}|}\}$. We prove Lemma 3 by proving the following two inequalities:

$$f(\mathcal{A}) \geq \sum_{i=1}^{|\mathcal{A}|} f(a_i | \mathcal{V} \setminus \{a_i\}), \tag{6.2}$$

$$\sum_{i=1}^{|\mathcal{A}|} f(a_i | \mathcal{V} \setminus \{a_i\}) \geq (1 - \kappa_f) \sum_{i=1}^{|\mathcal{A}|} f(a_i). \tag{6.3}$$

We begin with the proof of ineq. (6.2):

$$\begin{aligned} f(\mathcal{A}) &= f(\mathcal{A} | \emptyset) && (6.4)\\ &\geq f(\mathcal{A} | \mathcal{V} \setminus \mathcal{A}) && (6.5)\\ &= \sum_{i=1}^{|\mathcal{A}|} f(a_i | \mathcal{V} \setminus \{a_i, a_{i+1}, \ldots, a_{|\mathcal{A}|}\}) && (6.6)\\ &\geq \sum_{i=1}^{|\mathcal{A}|} f(a_i | \mathcal{V} \setminus \{a_i\}), && (6.7) \end{aligned}$$

where ineqs. (6.5) to (6.7) hold for the following reasons: ineq. (6.5) is implied by eq. (6.4) because f is submodular and $\emptyset \subseteq \mathcal{V} \setminus \mathcal{A}$; eq. (6.6) holds since for any sets $\mathcal{X} \subseteq \mathcal{V}$ and $\mathcal{Y} \subseteq \mathcal{V}$ we have $f(\mathcal{X} | \mathcal{Y}) = f(\mathcal{X} \cup \mathcal{Y}) - f(\mathcal{Y})$, and also $\{a_1, a_2, \ldots, a_{|\mathcal{A}|}\}$ denotes the set $\mathcal{A}$; and ineq. (6.7)

holds since f is submodular and $\mathcal{V} \setminus \{a_i, a_{i+1}, \ldots, a_\mu\} \subseteq \mathcal{V} \setminus \{a_i\}$. These observations complete the proof of ineq. (6.2).

We now prove ineq. (6.3) using the Definition 4 of κ_f, as follows: since $\kappa_f = 1 - \min_{v \in \mathcal{V}} \frac{f(v|\mathcal{V} \setminus \{v\})}{f(v)}$, it is implied that for all elements $v \in \mathcal{V}$ it is $f(v|\mathcal{V} \setminus \{v\}) \geq (1 - \kappa_f) f(v)$. Therefore, adding the latter inequality across all elements $a \in \mathcal{A}$ completes the proof of ineq. (6.3). ■

Lemma 2 Consider any finite set $\mathcal{V}$, a non-decreasing and submodular $f : 2^{\mathcal{V}} \mapsto \mathbb{R}$, and non-empty sets $\mathcal{Y}, \mathcal{P} \subseteq \mathcal{V}$ such that for all $y \in \mathcal{Y}$ and all $p \in \mathcal{P}$ $f(y) \geq f(p)$. Then:

$$f(\mathcal{P}|\mathcal{Y}) \leq |\mathcal{P}| f(\mathcal{Y}).$$

*Proof of Lemma 2 Consider any $y \in \mathcal{Y}$ (such an element exists since Lemma 2 considers that $\mathcal{Y}$ is non-empty); then,

$$\begin{aligned}
f(\mathcal{P}|\mathcal{Y}) &= f(\mathcal{P} \cup \mathcal{Y}) - f(\mathcal{Y}) && (6.8)\\
&\leq f(\mathcal{P}) + f(\mathcal{Y}) - f(\mathcal{Y}) && (6.9)\\
&= f(\mathcal{P}) \\
&\leq \sum_{p \in \mathcal{P}} f(p) && (6.10)\\
&\leq |\mathcal{P}| \max_{p \in \mathcal{P}} f(p) \\
&\leq |\mathcal{P}| f(y) && (6.11)\\
&\leq |\mathcal{P}| f(\mathcal{Y}), && (6.12)
\end{aligned}$$

where eq. (6.8) to ineq. (6.12) hold for the following reasons: eq. (6.8) holds since for any sets $\mathcal{X} \subseteq \mathcal{V}$ and $\mathcal{Y} \subseteq \mathcal{V}$, $f(\mathcal{X}|\mathcal{Y}) = f(\mathcal{X} \cup \mathcal{Y}) - f(\mathcal{Y})$; ineq. (6.9) holds since f is submodular and, as a result, the submodularity Definition 3 implies that for any set $\mathcal{A} \subseteq \mathcal{V}$ and $\mathcal{A}' \subseteq \mathcal{V}$, $f(\mathcal{A} \cup \mathcal{A}') \leq f(\mathcal{A}) + f(\mathcal{A}')$; ineq. (6.10) holds for the same reason as ineq. (6.9); ineq. (6.11)

holds since for all elements $y \in \mathcal{Y}$ and all elements $p \in \mathcal{P}$ $f(y) \geq f(p)$; finally, ineq. (6.12) holds because f is monotone and $y \in \mathcal{Y}$. ■

Lemma 3 Consider a finite set $\mathcal{V}$ and a non-decreasing $f : 2^{\mathcal{V}} \mapsto \mathbb{R}$ such that f is non-negative and $f(\emptyset) = 0$. For any set $\mathcal{A} \subseteq \mathcal{V}$ and any set $\mathcal{B} \subseteq \mathcal{V}$ such that $\mathcal{A} \cap \mathcal{B} = \emptyset$:

$$f(\mathcal{A} \cup \mathcal{B}) \geq (1 - c_f) \left(f(\mathcal{A}) + \sum_{b \in \mathcal{B}} f(b) \right).$$

*Proof of Lemma 3 Let $\mathcal{B} = \{b_1, b_2, \ldots, b_{|\mathcal{B}|}\}$. Then,

$$f(\mathcal{A} \cup \mathcal{B}) = f(\mathcal{A}) + \sum_{i=1}^{|\mathcal{B}|} f(b_i | \mathcal{A} \cup \{b_1, b_2, \ldots, b_{i-1}\}). \tag{6.13}$$

In addition, Definition 5 of total curvature implies:

$$\begin{aligned} f(b_i | \mathcal{A} \cup \{b_1, b_2, \ldots, b_{i-1}\}) &\geq (1 - c_f) f(b_i | \emptyset) \\ &= (1 - c_f) f(b_i), \end{aligned} \tag{6.14}$$

where the latter equation holds since $f(\emptyset) = 0$. The proof is completed by substituting (6.14) in (6.13) and then taking into account that $f(\mathcal{A}) \geq (1 - c_f) f(\mathcal{A})$ since $0 \leq c_f \leq 1$. ■

Lemma 4 Consider a finite set $\mathcal{V}$ and a non-decreasing $f : 2^{\mathcal{V}} \mapsto \mathbb{R}$ such that f is non-negative and $f(\emptyset) = 0$. For any $\mathcal{A} \subseteq \mathcal{V}$ and any $\mathcal{B} \subseteq \mathcal{V}$ such that $\mathcal{A} \setminus \mathcal{B} \neq \emptyset$:

$$f(\mathcal{A}) + (1 - c_f) f(\mathcal{B}) \geq (1 - c_f) f(\mathcal{A} \cup \mathcal{B}) + f(\mathcal{A} \cap \mathcal{B}).$$

*Proof of Lemma 4 Let $\mathcal{A} \setminus \mathcal{B} = \{i_1, i_2, \ldots, i_r\}$, where $r = |\mathcal{A} - \mathcal{B}|$. From Definition 5 of total curvature c_f, for any $i = 1, 2, \ldots, r$, it is $f(i_j | \mathcal{A} \cap \mathcal{B} \cup \{i_1, i_2, \ldots, i_{j-1}\}) \geq (1 - c_f) f(i_j | \mathcal{B} \cup$

$\{i_1, i_2, \ldots, i_{j-1}\}$). Summing these r inequalities,

$$f(\mathcal{A}) - f(\mathcal{A} \cap \mathcal{B}) \geq (1 - c_f)\left(f(\mathcal{A} \cup \mathcal{B}) - f(\mathcal{B})\right),$$

which implies the lemma. ■

Corollary 1 Consider a finite set $\mathcal{V}$ and a non-decreasing $f : 2^{\mathcal{V}} \mapsto \mathbb{R}$ such that f is non-negative and $f(\emptyset) = 0$. For any $\mathcal{A} \subseteq \mathcal{V}$ and any $\mathcal{B} \subseteq \mathcal{V}$ such that $\mathcal{A} \cap \mathcal{B} = \emptyset$:

$$f(\mathcal{A}) + \sum_{b \in \mathcal{B}} f(b) \geq (1 - c_f) f(\mathcal{A} \cup \mathcal{B}).$$

*Proof of Corollary 1 Let $\mathcal{B} = \{b_1, b_2, \ldots, b_{|\mathcal{B}|}\}$.

$$\begin{aligned} f(\mathcal{A}) + \sum_{i=1}^{|\mathcal{B}|} f(b_i) &\geq (1 - c_f) f(\mathcal{A}) + \sum_{i=1}^{|\mathcal{B}|} f(b_i)) \qquad (6.15) \\ &\geq (1 - c_f) f(\mathcal{A} \cup \{b_1\}) + \sum_{i=2}^{|\mathcal{B}|} f(b_i) \\ &\geq (1 - c_f) f(\mathcal{A} \cup \{b_1, b_2\}) + \sum_{i=3}^{|\mathcal{B}|} f(b_i) \\ &\vdots \\ &\geq (1 - c_f) f(\mathcal{A} \cup \mathcal{B}), \end{aligned}$$

where (6.15) holds since $0 \leq c_f \leq 1$, and the rest due to Lemma 4 since $\mathcal{A} \cap \mathcal{B} = \emptyset$ implies $\mathcal{A} \setminus \{b_1\} \neq \emptyset$, $\mathcal{A} \cup \{b_1\} \setminus \{b_2\} \neq \emptyset$, …, $\mathcal{A} \cup \{b_1, b_2, \ldots, b_{|\mathcal{B}|-1}\} \setminus \{b_{|\mathcal{B}|}\} \neq \emptyset$.

■

B Coordinate Descent

We describe coordinate descent [4, Section IV], and generalize the proof in [4] that coordinate descent guarantees an approximation performance up to a multiplicative factor 1/2 the

optimal when the information objective function is the mutual information. In particular, we extend the proof to any non-decreasing and possibly submodular information objective function; the result will support the proof of Proposition 1.

The algorithm coordinate descent works as follows: consider an arbitrary ordering of the robots in $\mathcal{V}$, such that $\mathcal{V} \equiv \{1,2,\ldots,n\}$, and suppose that robot 1 first chooses its controls, without considering the other robots; in other words, robot 1 solves the single robot version of Problem 7, i.e., $J_{\{1\},t+1:t+\mathsf{T}_{\mathsf{PLAN}}} := J(u_1)$, to obtain controls $u_{\{1\}}$ such that:

$$u^{cd}_{1,\,t+1:t+\mathsf{T}_{\mathsf{PLAN}}} = \arg \max_{\substack{u_{i,t'} \in \mathcal{U}_{i,t'} \\ t'=t+1:t+\mathsf{T}_{\mathsf{PLAN}}}} J(u_{i,\,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}) \tag{6.16}$$

Afterwards, robot 1 communicates its chosen control sequence to robot 2, and robot 2, given the control sequence of robot 1.computes its control input as follows, assuming the control inputs for robot 1 are fixed:

$$u^{cd}_{2,\,t+1:t+\mathsf{T}_{\mathsf{PLAN}}} = \arg \max_{\substack{u_{i,t'} \in \mathcal{U}_{i,t'} \\ t'=t+1:t+\mathsf{T}_{\mathsf{PLAN}}}} J(u^{cd}_1, u_{i,\,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}) \tag{6.17}$$

This continues such that robot i solves a single robot problem, given the control inputs from the robots $1,2,\ldots,i-1$:

$$u^{cd}_{i,\,t+1:t+\mathsf{T}_{\mathsf{PLAN}}} = \arg \max_{\substack{u_{i,t'} \in \mathcal{U}_{i,t'} \\ t'=t+1:t+\mathsf{T}_{\mathsf{PLAN}}}} J(u^{cd}_{1:i-1}, u_{i,\,t+1:t+\mathsf{T}_{\mathsf{PLAN}}}) \tag{6.18}$$

Notably, if we let u^*_i be the control inputs for the i-th robot resulting from the optimal solution to the n robot problem, then from the coordinate descent algorithm, we have:

$$J(u^{cd}_{1:i-1}, u^\star_i) \leq J(u^{cd}_{1:i}) \tag{6.19}$$

Lemma 5 (Approximation performance of coordinate descent) Consider a set of robots $\mathcal{V}$, and an instance of problem (P-RTP). Denote the optimal control inputs for problem (P-RTP),

across all robots and all times, by $u^{\star}_{\mathcal{V},t+1:t+\mathsf{T}_{\mathsf{PLAN}}}$. The coordinate descent algorithm returns control inputs $u^{cd}_{\mathcal{V},t+1:t+\mathsf{T}_{\mathsf{PLAN}}}$, across all robots and all times, such that:

- if the objective function J is non-decreasing submodular in the active robot set, and (without loss of generality) J is non-negative and $J(\emptyset) = 0$, then:

$$\frac{J(u^{cd}_{1:n})}{J(u^{\star}_{1:n})} \geq \frac{1}{2}. \tag{6.20}$$

- If the objective function J is non-decreasing in the active robot set, and (without loss of generality) J is non-negative and $J(\emptyset) = 0$, then:

$$\frac{J(u^{cd}_{1:n})}{J(u^{\star}_{1:n})} \geq \frac{1 - c_J}{2}. \tag{6.21}$$

*Proof of Lemma 5

- if the objective function J is non-decreasing and submodular in the active robot set, and (without loss of generality) J is non-negative and $J(\emptyset) = 0$, then:

$$\begin{aligned}
J(u^{\star}_{1:n}) &\leq J(u^{\star}_{1:n}) + \sum_{i=1}^{n} [J(u^{cd}_{1:i}, u^{\star}_{i+1:n}) && (6.22)\\
&\qquad - J(u^{cd}_{1:i-1}, u^{\star}_{i+1:n})] \\
&= J(u^{cd}_{1:n}) + \sum_{i=1}^{n} [J(u^{cd}_{1:i-1}, u^{\star}_{i:n}) && (6.23)\\
&\qquad - J(u^{cd}_{1:i-1}, u^{\star}_{i+1:n})] \\
&= J(u^{cd}_{1:n}) + \sum_{i=1}^{n} J(u^{\star}_{i} | \{u^{cd}_{1:i-1}, u^{\star}_{i+1,n}\}) && (6.24)\\
&\leq J(u^{cd}_{1:n}) + \sum_{i=1}^{n} J(u^{\star}_{i} | u^{cd}_{1:i-1}) && (6.25)\\
&\leq J(u^{cd}_{1:n}) + \sum_{i=1}^{n} J(u^{cd}_{i} | u^{cd}_{1:i-1}) && (6.26)
\end{aligned}$$

$$= J(u^{cd}_{1:n}) + J(u^{cd}_{1:n}) \tag{6.27}$$

$$\leq 2J(u^{cd}_{1:n}), \tag{6.28}$$

where ineq. (6.22) holds due to monotonicity of J; eq. 6.23) is a shift in indexes of the first term in the sum; eq. (6.24) is an expression of the sum as a sum of marginal gains; ineq. (6.25) holds due to submodularity; ineq. (6.26) holds by the coordinate-descent policy (per eq. (6.19)); eq. (6.27) holds due to the definition of the marginal gain symbol $J(u^{\star}_i|u^{cd}_{1:i-1})$ (for any $i = 1,2,\ldots,n$) as $J(u^{\star}_i, u^{cd}_{1:i-1}) - J(u^{cd}_{1:i-1})$; finally, a re-arrangement of the terms in eq. (6.28) gives $J(u^{cd}_{1:n})/J(u^{*}_{1:n}) \geq 1/2$.

- If J is non-decreasing in the active robot set, and (without loss of generality) J is non-negative and $J(\emptyset) = 0$, then multiplying both sides of eq. (6.24) (which holds for any non-decreasing J) with $(1-c_J)$, we have:

$$\begin{aligned}
&(1-c_J)J(u^{\star}_{1:n}) \\
&\quad = (1-c_J)J(u^{cd}_{1:n}) + \\
&\qquad\qquad (1-c_J)\sum_{i=1}^{n} J(u^{\star}_i|\{u^{cd}_{1:i-1}, u^{\star}_{i+1,n}\}) \\
&\quad \leq J(u^{cd}_{1:n}) + (1-c_J)\sum_{i=1}^{n} J(u^{\star}_i|\{u^{cd}_{1:i-1}, u^{\star}_{i+1,n}\}) && (6.29)\\
&\quad \leq J(u^{cd}_{1:n}) + \sum_{i=1}^{n} J(u^{\star}_i|u^{cd}_{1:i-1}) && (6.30)\\
&\quad \leq J(u^{cd}_{1:n}) + \sum_{i=1}^{n} J(u^{cd}_i|u^{cd}_{1:i-1}) && (6.31)\\
&\quad = J(u^{cd}_{1:n}) + J(u^{cd}_{1:n}) && (6.32)\\
&\quad \leq 2J(u^{cd}_{1:n}), && (6.33)
\end{aligned}$$

where, ineq. (6.29) holds since $0 \leq c_J \leq 1$; ineq. (6.30) holds since J is non-decreasing in the set of active robots, and Definition 5 of total curvature implies that for any non-decreasing set function $g : 2^{\mathcal{V}} \mapsto \mathbb{R}$, for any element $v \in \mathcal{V}$, and for any set $\mathcal{A}, \mathcal{B} \subseteq \mathcal{V} \setminus \{v\}$:

$$(1 - c_g) g(v|\mathcal{B}) \leq g(\{v\}|\mathcal{A}); \tag{6.34}$$

ineq. (6.31) holds by the coordinate-descent algorithm; eq. (6.32) holds due to the definition of the marginal gain symbol $J(u_i^\star | u_{1:i-1}^{cd})$ (for any $i = 1, 2, \ldots, n$) as $J(u_i^\star, u_{1:i-1}^{cd}) - J(u_{1:i-1}^{cd})$; finally, a re-arrangement of terms gives $J(u_{1:n}^{cd})/J(u_{1:n}^*) \geq (1 - c_J)/2$. ■

C Proof of Theorem 1

We first prove Theorem 1's part 1 (approximation performance), and then, Theorem 1's part 2 (running time).

C.1 Proof of Theorem 1's Part 1 (Approximation Performance)

The proof follows the steps of the proof of [85, Theorem 1] and [100, Theorem 1]. We first prove eq. (4.9), then, eq. (4.8).

To the above ends, we use the following notation (along with the notation introduced in Theorem 1 and in Appendix A): given that using Algorithm 10 the robots in $\mathcal{V}$ select control inputs $u_{\mathcal{V}, t+1:t+\mathsf{T}_{\mathsf{PLAN}}}$, then, for notational simplicity:

- let $\mathcal{A}^\star \triangleq \mathcal{A}^\star(u_{\mathcal{V}, t+1:t+\mathsf{T}_{\mathsf{PLAN}}})$;
- let $\mathcal{L}^+ \triangleq \mathcal{L} \setminus \mathcal{A}^\star$, i.e., $\mathcal{S}_1$ be the remaining robots in $\mathcal{L}$ after the removal of the robots in $\mathcal{A}^\star$;
- let $(\mathcal{V} \setminus \mathcal{L})^+ \triangleq (\mathcal{V} \setminus \mathcal{L}) \setminus \mathcal{A}^\star$, i.e., $\mathcal{S}_2$ be the remaining robots in $\mathcal{V} \setminus \mathcal{L}$ after the removal of the robots in $\mathcal{A}^\star$.

*Proof of ineq. (4.9) The proof follows the steps of the proof of [85, Theorem 1]. Consider that the objective function J is non-decreasing and submodular in the active robot set, such that (without loss of generality) J is non-negative and $J(\emptyset) = 0$. We first prove the part $1 - \kappa_J$ of the bound in the right-hand-side of ineq. (4.9), and then, the part $h(|\mathcal{V}|, \alpha)$ of the bound in the right-hand-side of ineq. (4.9).

To prove the part $1 - \kappa_J$ of the bound in the right-hand-side of ineq. (4.9), we follow the steps of the proof of [85, Theorem 1], and make the following observations:

$$
\begin{aligned}
& J(\mathcal{V} \setminus \mathcal{A}^\star) \\
& \quad = J(\mathcal{L}^+ \cup (\mathcal{V} \setminus \mathcal{L})^+) && (6.35) \\
& \quad \geq (1 - \kappa_J) \sum_{v \in \mathcal{L}^+ \cup (\mathcal{V} \setminus \mathcal{L})^+} J(v) && (6.36) \\
& \quad \geq (1 - \kappa_J) \left(\sum_{v \in (\mathcal{V} \setminus \mathcal{L}) \setminus (\mathcal{V} \setminus \mathcal{L})^+} J(v) + \sum_{v \in (\mathcal{V} \setminus \mathcal{L})^+} J(v) \right) && (6.37) \\
& \quad \geq (1 - \kappa_J) J\{[(\mathcal{V} \setminus \mathcal{L}) \setminus (\mathcal{V} \setminus \mathcal{L})^+] \cup (\mathcal{V} \setminus \mathcal{L})^+\} && (6.38) \\
& \quad = (1 - \kappa_J) J(\mathcal{V} \setminus \mathcal{L}), && (6.39)
\end{aligned}
$$

where eq. (6.35) to (6.39) hold for the following reasons: eq. (6.35) follows from the definitions of the sets $\mathcal{L}^+$ and $(\mathcal{V} \setminus \mathcal{L})^+$; ineq. (6.36) follows from ineq. (6.35) due to Lemma 1; ineq. (6.37) follows from ineq. (6.36) because for all elements $v \in \mathcal{L}^+$ and all elements $v' \in (\mathcal{V} \setminus \mathcal{L}) \setminus (\mathcal{V} \setminus \mathcal{L})^+$ we have $J(v) \geq J(v')$ (note that due to the definitions of the sets $\mathcal{L}^+$ and $(\mathcal{V} \setminus \mathcal{L})^+$, $|\mathcal{L}^+| = |(\mathcal{V} \setminus \mathcal{L}) \setminus (\mathcal{V} \setminus \mathcal{L})^+|$, that is, the number of non-removed elements in $\mathcal{L}$ is equal to the number of removed elements in $\mathcal{V} \setminus \mathcal{L}$); finally, ineq. (6.38) follows from ineq. (6.37) because the set function J is submodular and, as a result, the submodularity Definition 3 implies that for any sets $\mathcal{S} \subseteq \mathcal{V}$ and $\mathcal{S}' \subseteq \mathcal{V}$, $J(\mathcal{S}) + J(\mathcal{S}') \geq J(\mathcal{S} \cup \mathcal{S}')$ [91, Proposition 2.1]. We now complete the proof of the part $1 - \kappa_J$ of the bound in the right-hand-side of ineq. (4.9) by proving

that in ineq. (6.39):

$$J(\mathcal{V} \setminus \mathcal{L}) \geq J^{\star}, \tag{6.40}$$

when the robots in $\mathcal{V}$ optimally solve the problems in Algorithm 10's step 6, per the statement of Theorem 1. In particular, if for any active robot set $\mathcal{R} \subseteq \mathcal{V}$, we let $\bar{u}_{\mathcal{R}} \triangleq \{\bar{u}_{i,t'} : \ \bar{u}_{i,t'} \in \mathcal{U}_{i,t'}, \ \ i \in \mathcal{R}, \ \ t' = t+1, \ldots, t+\mathsf{T}_{\mathsf{PLAN}}\}$ denote a collection of control inputs to the robots in $\mathcal{R}$, then:

$$\begin{aligned} J(\mathcal{V} \setminus \mathcal{L}) \equiv & \max_{\substack{\bar{u}_{i,t} \in \mathcal{U}_{i,t}, i \in \mathcal{V}, \\ t' = t+1 : t+\mathsf{T}_{\mathsf{PLAN}}}} J(\bar{u}_{\mathcal{V} \setminus \mathcal{L}, t+1:t+\mathsf{T}_{\mathsf{PLAN}}}) && (6.41) \\ \geq & \min_{\substack{\bar{\mathcal{L}} \subseteq \mathcal{V}, \\ |\bar{\mathcal{L}}| \leq \alpha}} \max_{\substack{\bar{u}_{i,t} \in \mathcal{U}_{i,t}, i \in \mathcal{V}, \\ t' = t+1 : t+\mathsf{T}_{\mathsf{PLAN}}}} J(\bar{u}_{\mathcal{V} \setminus \bar{\mathcal{L}}, t+1:t+\mathsf{T}_{\mathsf{PLAN}}}) && (6.42) \\ \geq & \max_{\substack{\bar{u}_{i,t} \in \mathcal{U}_{i,t}, i \in \mathcal{V}, \\ t = t+1 : t+\mathsf{T}_{\mathsf{PLAN}}}} \min_{\substack{\bar{\mathcal{L}} \subseteq \mathcal{V}, \\ |\bar{\mathcal{L}}| \leq \alpha}} J(\bar{u}_{\mathcal{V} \setminus \bar{\mathcal{L}}, t+1:t+\mathsf{T}_{\mathsf{PLAN}}}) && (6.43) \\ \equiv & J^{\star}, && (6.44) \end{aligned}$$

where (6.41)-(6.44) hold true since: the equivalence in eq. (6.41) holds since the robots in $\mathcal{V}$ solve optimally the problems in Algorithm 10's step 6, per the statement of Theorem 1; (6.42) holds since we minimize over the set $\mathcal{L}$; (6.43) holds because for any set $\hat{\mathcal{L}} \subseteq \mathcal{V}$ and any control inputs $\hat{u}_{\mathcal{R},t+1:t+\mathsf{T}_{\mathsf{PLAN}}} \triangleq \{\hat{u}_{i,t} : \ \ \hat{u}_{i,t} \in \mathcal{U}_{i,t}, \ \ i \in \mathcal{R}, \ \ t' = t+1, \ldots, t+\mathsf{T}_{\mathsf{PLAN}}\}$:

$$\max_{\substack{\bar{u}_{i,t} \in \mathcal{U}_{i,t}, i \in \mathcal{V}, \\ t' = t+1 : t+\mathsf{T}_{\mathsf{PLAN}}}} J(\bar{u}_{\mathcal{V} \setminus \bar{\mathcal{L}}, t+1:t+\mathsf{T}_{\mathsf{PLAN}}}) \geq J(\hat{u}_{\mathcal{V} \setminus \hat{\mathcal{L}}, t+1:t+\mathsf{T}_{\mathsf{PLAN}}}),$$

which implies

$$\min_{\substack{\bar{\mathcal{L}} \subseteq \mathcal{V}, \\ |\bar{\mathcal{L}}| \leq \alpha}} \max_{\substack{\bar{u}_{i,t} \in \mathcal{U}_{i,t}, i \in \mathcal{V}, \\ t' = t+1 : t+\mathsf{T}_{\mathsf{PLAN}}}} J(\bar{u}_{\mathcal{V} \setminus \bar{\mathcal{L}}, t+1:t+\mathsf{T}_{\mathsf{PLAN}}}) \geq \min_{\substack{\bar{\mathcal{L}} \subseteq \mathcal{V}, \\ |\bar{\mathcal{L}}| \leq \alpha}} J(\hat{u}_{\mathcal{V} \setminus \bar{\mathcal{L}}, t+1:t+\mathsf{T}_{\mathsf{PLAN}}}) \Rightarrow$$

$$\min_{\substack{\bar{\mathcal{L}} \subseteq \mathcal{V}, \\ |\bar{\mathcal{L}}| \leq \alpha}} \max_{\substack{\bar{u}_{i,t} \in \mathcal{U}_{i,t}, i \in \mathcal{V}, \\ t' = t+1 : t+\mathsf{T}_{\mathsf{PLAN}}}} J(\bar{u}_{\mathcal{V} \setminus \bar{\mathcal{L}}, t+1:t+\mathsf{T}_{\mathsf{PLAN}}}) \geq \max_{\substack{\bar{u}_{i,t} \in \mathcal{U}_{i,t}, i \in \mathcal{V}, \\ t' = t+1 : t+\mathsf{T}_{\mathsf{PLAN}}}} \min_{\substack{\bar{\mathcal{L}} \subseteq \mathcal{V}, \\ |\bar{\mathcal{L}}| \leq \alpha}} J(\bar{u}_{\mathcal{V} \setminus \bar{\mathcal{L}}, t+1:t+\mathsf{T}_{\mathsf{PLAN}}}),$$

where the last one is eq. (6.43); finally, the equivalence in eq. (6.44) holds since $J^\star$ (per the statement of Theorem 1) denotes the optimal value to Problem 7. Overall, we proved that ineq. (6.44) proves ineq. (6.40); and, now, the combination of ineq. (6.39) and ineq. (6.40) proves the part $1 - \kappa_J$ of the bound in the right-hand-side of ineq. (4.9).

We finally prove the part $1/(1+\alpha)$ of the bound in the right-hand-side of ineq. (4.9), and complete this way the proof of Theorem 1. To this end, we follow the steps of the proof of [85, Theorem 1], and use the notation introduced in Fig. 6.1, along with the following notation:

$$\eta \triangleq \frac{J(\mathcal{A}_2^\star | \mathcal{V} \setminus \mathcal{A}^\star)}{J(\mathcal{V} \setminus \mathcal{L})} \tag{6.45}$$

Later in this proof, we prove $0 \leq \eta \leq 1$. We first observe that:

$$J(\mathcal{V} \setminus \mathcal{A}^\star) \geq \max\{J(\mathcal{V} \setminus \mathcal{A}^\star), J(\mathcal{L}^+)\}; \tag{6.46}$$

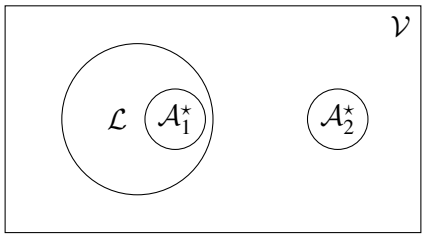


Figure 6.1. Venn diagram, where the set $\mathcal{L}$ is the robot set defined in step 3 of Algorithm 10, and the set $\mathcal{A}_1^\star$ and the set $\mathcal{A}_2^\star$ are such that $\mathcal{A}_1^\star = \mathcal{A}^\star \cap \mathcal{L}$, and $\mathcal{A}_2^\star = \mathcal{A}^\star \cap (\mathcal{V} \setminus \mathcal{L})$ (observe that these definitions imply $\mathcal{A}_1^\star \cap \mathcal{A}_2^\star = \emptyset$ and $\mathcal{A}^\star = \mathcal{A}_1^\star \cup \mathcal{A}_2^\star$).

in the following paragraphs, we prove the three inequalities:

$$J(\mathcal{V} \setminus \mathcal{A}^\star) \geq (1-\eta) J(\mathcal{V} \setminus \mathcal{L}), \tag{6.47}$$

$$J(\mathcal{L}^+) \geq \eta \frac{1}{\alpha} J(\mathcal{V} \setminus \mathcal{L}), \tag{6.48}$$

$$\max\{(1-\eta), \eta \frac{1}{\alpha}\} \geq \frac{1}{\alpha+1}. \tag{6.49}$$

Then, if we substitute ineq. (6.47), ineq. (6.48) and ineq. (6.49) to ineq. (6.46), and take into account that $J(\mathcal{V} \setminus \mathcal{L}) \geq 0$, then:

$$J(\mathcal{V} \setminus \mathcal{A}^\star) \geq \frac{1}{\alpha+1} J(\mathcal{V} \setminus \mathcal{L}),$$

which implies the part $1/(1+\alpha)$ of the bound in the right-hand-side of ineq. (4.9), after taking into account ineq. (6.40).

We next complete the proof of the part $1/(1+\alpha)$ of the bound in the right-hand-side of ineq. (4.9) by proving $0 \leq \eta \leq 1$, ineq. (6.47), ineq. (6.48), and ineq. (6.49).

Proof of ineq. $0 \leq \eta \leq 1$ We first prove $\eta \geq 0$, and then $\eta \leq 1$: $\eta \geq 0$, since $\eta \equiv J(\mathcal{A}_2^\star | \mathcal{V} \setminus$

$\mathcal{A}^\star)/J(\mathcal{V}\setminus\mathcal{L})$, and J is non-negative; and $\eta \leq 1$, since $J(\mathcal{V}\setminus\mathcal{L}) \geq J(\mathcal{A}_2^\star)$, due to monotonicity of J and that $\mathcal{A}_2^\star \subseteq \mathcal{V}\setminus\mathcal{L}$, and $J(\mathcal{A}_2^\star) \geq J(\mathcal{A}_2^\star|\mathcal{V}\setminus\mathcal{A}^\star)$, due to submodularity of J and that $\emptyset \subseteq \mathcal{V}\setminus\mathcal{A}^\star$.

Proof of ineq. (6.47) We complete the proof of ineq. (6.47) in two steps. First, it can be verified that:

$$\begin{aligned} f(\mathcal{V}\setminus\mathcal{A}^\star) &= f(\mathcal{V}\setminus\mathcal{L}) - \\ & J(\mathcal{A}_2^\star|\mathcal{V}\setminus\mathcal{A}^\star) + J(\mathcal{L}|\mathcal{V}\setminus\mathcal{L}) - J(\mathcal{A}_1^\star|\mathcal{V}\setminus\mathcal{A}_1^\star), \end{aligned} \tag{6.50}$$

since for any sets $\mathcal{X} \subseteq \mathcal{V}$ and $\mathcal{Y} \subseteq \mathcal{V}$, $J(\mathcal{X}|\mathcal{Y}) \equiv J(\mathcal{X}\cup\mathcal{Y}) - J(\mathcal{Y})$. Second, eq. (6.50) implies ineq. (6.47), since $J(\mathcal{A}_2^\star|\mathcal{V}\setminus\mathcal{A}^\star) = \eta J(\mathcal{V}\setminus\mathcal{L})$, and $J(\mathcal{L}|\mathcal{V}\setminus\mathcal{L}) - J(\mathcal{A}_1^\star|\mathcal{V}\setminus\mathcal{A}_1^\star) \geq 0$; the latter is true due to the following two observations: $J(\mathcal{L}|\mathcal{V}\setminus\mathcal{L}) \geq J(\mathcal{A}_1^\star|\mathcal{V}\setminus\mathcal{L})$, since J is monotone and $\mathcal{A}_1^\star \subseteq \mathcal{L}$; and $J(\mathcal{A}_1^\star|\mathcal{V}\setminus\mathcal{L}) \geq J(\mathcal{A}_1^\star|\mathcal{V}\setminus\mathcal{A}_1^\star)$, since J is submodular and $\mathcal{V}\setminus\mathcal{L} \subseteq \mathcal{V}\setminus\mathcal{A}_1^\star$ (see also Fig. 6.1).

Proof of ineq. (6.48) To prove ineq. (6.48), since $\mathcal{A}_2^\star \neq \emptyset$ (and, as a result, also $\mathcal{L}^+ \neq \emptyset$), and for all elements $a \in \mathcal{L}^+$ and all elements $b \in \mathcal{A}_2^\star$, $J(a) \geq J(b)$, from Lemma 2 we have:

$$\begin{aligned} J(\mathcal{A}_2^\star|\mathcal{L}^+) &\leq |\mathcal{A}_2^\star| J(\mathcal{L}^+) \\ &\leq \alpha J(\mathcal{L}^+), \end{aligned} \tag{6.51}$$

since $|\mathcal{A}_2^\star| \leq \alpha$. Overall,

$$J(\mathcal{L}^+) \geq \frac{1}{\alpha} J(\mathcal{A}_2^\star|\mathcal{L}^+) \tag{6.52}$$

$$\geq \frac{1}{\alpha} J(\mathcal{A}_2^\star|\mathcal{L}^+ \cup (\mathcal{V}\setminus\mathcal{L})^+) \tag{6.53}$$

$$= \frac{1}{\alpha} J(\mathcal{A}_2^\star | \mathcal{V} \setminus \mathcal{A}^\star) \tag{6.54}$$

$$= \eta \frac{1}{\alpha} J(\mathcal{V} \setminus \mathcal{L}), \tag{6.55}$$

where ineq. (6.52) to eq. (6.55) hold for the following reasons: ineq. (6.52) follows from ineq. (6.51); ineq. (6.53) holds since J is submodular and $\mathcal{L}^+ \subseteq \mathcal{L}^+ \cup (\mathcal{V} \setminus \mathcal{L})^+$; eq. (6.54) holds due to the definitions of the sets $\mathcal{L}^+$, $(\mathcal{V} \setminus \mathcal{L})^+$ and $\mathcal{A}^\star$; finally, eq. (6.55) holds due to the definition of η. Overall, the latter derivation concludes the proof of ineq. (6.48).

Proof of ineq. (6.49) Let $b = 1/\alpha$. We complete the proof first for the case where $(1-\eta) \geq \eta b$, and then for the case $(1-\eta) < \eta b$: i) When $(1-\eta) \geq \eta b$, $\max\{(1-\eta), \eta b\} = 1-\eta$ and $\eta \leq 1/(1+b)$. Due to the latter, $1-\eta \geq b/(1+b) = 1/(\alpha+1)$ and, as a result, (6.49) holds. ii) When $(1-\eta) < \eta b$, $\max\{(1-\eta), \eta b\} = \eta b$ and $\eta > 1/(1+b)$. Due to the latter, $\eta b > b/(1+b)$ and, as a result, (6.49) holds.

We completed the proof of $0 \leq \eta \leq 1$, and of ineqs. (6.47), (6.48) and (6.49). Thus, we also completed the proof of the part $1/(1+\alpha)$ of the bound in the right-hand-side of ineq. (4.9), and, in sum, the proof of ineq. (4.9).

*Proof of ineq. (4.8) Consider that the objective function J is non-decreasing in the active robot set, such that (without loss of generality) J is non-negative and $J(\emptyset) = 0$.

The proof follows the steps of the proof of [100, Theorem 1], by making the following

observations:

$$
\begin{aligned}
& J(\mathcal{V} \setminus \mathcal{A}^\star) \\
& \quad = J(\mathcal{L}^+ \cup (\mathcal{V} \setminus \mathcal{L})^+) && (6.56) \\
& \quad \geq (1 - c_J) \sum_{v \in \mathcal{L}^+ \cup (\mathcal{V} \setminus \mathcal{L})^+} J(v) && (6.57) \\
& \quad \geq (1 - c_J) \left(\sum_{v \in (\mathcal{V} \setminus \mathcal{L}) \setminus (\mathcal{V} \setminus \mathcal{L})^+} J(v) + \sum_{v \in (\mathcal{V} \setminus \mathcal{L})^+} J(v) \right) && (6.58) \\
& \quad \geq (1 - c_J)^2 J\{[(\mathcal{V} \setminus \mathcal{L}) \setminus (\mathcal{V} \setminus \mathcal{L})^+] \cup (\mathcal{V} \setminus \mathcal{L})^+\} && (6.59) \\
& \quad = (1 - c_J)^2 J(\mathcal{V} \setminus \mathcal{L}), && (6.60)
\end{aligned}
$$

where eq. (6.56) to (6.60) hold for the following reasons: eq. (6.56) follows from the definitions of the sets $\mathcal{L}^+$ and $(\mathcal{V} \setminus \mathcal{L})^+$; ineq. (6.57) follows from ineq. (6.56) due to Lemma 3; ineq. (6.58) follows from ineq. (6.57) because for all elements $v \in \mathcal{L}^+$ and all elements $v' \in (\mathcal{V} \setminus \mathcal{L}) \setminus (\mathcal{V} \setminus \mathcal{L})^+$ we have $J(v) \geq J(v')$ (note that due to the definitions of the sets $\mathcal{L}^+$ and $(\mathcal{V} \setminus \mathcal{L})^+$ it is $|\mathcal{L}^+| = |(\mathcal{V} \setminus \mathcal{L}) \setminus (\mathcal{V} \setminus \mathcal{L})^+|$, that is, the number of non-removed elements in $\mathcal{L}$ is equal to the number of removed elements in $\mathcal{V} \setminus \mathcal{L}$); finally, ineq. (6.59) follows from ineq. (6.58) because the set function J is non-decreasing and Corollary 1 applies. Overall, the combination of ineq. (6.60) and ineq. (6.40) (observe that ineq. (6.40) still holds if the objective function J is merely non-decreasing) proves ineq. (4.8). ■

C.2 Proof of Theorem 1's Part 2 (Running Time)

RTP 's running time is found by adding the running time of (i) lines 1-3, i.e., $|\mathcal{V}|\rho$, (ii) line 4, i.e., $|\mathcal{V}| \log(|\mathcal{V}|)$ (using, e.g., the merge sort algorithm), (iii) lines 5-7, whose running time can be ignored since the optimization problems in line 6 have already been solved in lines 1-3, and (iv) line 8, i.e., ρ. The total is $|\mathcal{V}|(\rho + 1) + |\mathcal{V}| \log(|\mathcal{V}|) = O(|\mathcal{V}|\rho)$. ■

D Proof of Proposition 1

We first prove Proposition 1's part 1 (approximation bounds), and then, Proposition 1's part 2 (running time).

D.1 Proof of Proposition 1's Part 1 (Approximation Bounds)

The proof follows the steps of the proof of Theorem 1; hence, we describe here only the steps where the proof differs.

We first prove ineq. (4.11); then, we prove ineq. (4.10).

*Proof of ineq. (4.11) Consider that the objective function J is non-decreasing and submodular in the active robot set, such that (without loss of generality) J is non-negative and $J(\emptyset) = 0$. Since, per Proposition 1, Algorithm 10 calls the coordinate descent algorithm in step 4, the equivalence in eq. (6.41) is now invalid, and, in particular, using Lemma 5, the following inequality holds instead:

$$J(\mathcal{V} \setminus \mathcal{L}) \geq \frac{1}{2} \max_{\substack{\bar{u}_{i,t} \in \mathcal{U}_{i,t}, i \in \mathcal{V}, \\ t' = t+1 : t+\mathsf{T}_{\mathsf{PLAN}}}} J(\bar{u}_{\mathcal{V} \setminus \mathcal{L}, t+1:t+\mathsf{T}_{\mathsf{PLAN}}}). \tag{6.61}$$

Using ineq. (6.61), and following the same steps as in eqs. (6.41)-(6.44), we conclude:

$$J(\mathcal{V} \setminus \mathcal{L}) \geq \frac{1}{2} J^{\star}. \tag{6.62}$$

Using ineq. (6.62) the same way that ineq. (6.40) was used in the proof of Theorem 1's part 1, ineq. (4.10) is proved.

*Proof of ineq. (4.10) Consider that the objective function J is non-decreasing in the active robot set, such that (without loss of generality) J is non-negative and $J(\emptyset) = 0$. Similarly with the observations we made in the proof of ineq. (4.11), since, per Proposition 1, Algo-

rithm 10 calls the coordinate descent algorithm in step 4, the equivalence in eq. (6.41) is now invalid, and, in particular, using Lemma 5, the following inequality holds instead:

$$J(\mathcal{V}\setminus\mathcal{L}) \geq \frac{1-c_J}{2} \max_{\substack{\bar{u}_{i,t}\in\mathcal{U}_{i,t}, i\in\mathcal{V},\\ t'=t+1:t+\mathsf{T}_{\mathsf{PLAN}}}} J(\bar{u}_{\mathcal{V}\setminus\mathcal{L},t+1:t+\mathsf{T}_{\mathsf{PLAN}}}). \tag{6.63}$$

Using ineq. (6.63), and following the same steps as in eqs. (6.41)-(6.44), we conclude:

$$J(\mathcal{V}\setminus\mathcal{L}) \geq \frac{1-c_J}{2} J^{\star}. \tag{6.64}$$

Using ineq. (6.64) the same way that ineq. (6.40) was used in the proof of Theorem 1's part 1, ineq. (4.10) is proved. ■

D.2 Proof of Proposition 1's Part 2 (Communication Rounds)

Per B, coordinate descent needs to solve $|\mathcal{V}|$ times an optimization problem of same complexity as line 2 of RTP . RTP , instead, needs to solve $|\mathcal{V}|$ times the optimization problem in line 2 (per the "for" loop in lines 1-3), and the optimization problem in line 8, using coordinate descent. Since the running time of line 4 and lines 5-7 is negligible (cf. C.2) and can be ignored, the total running time of RTP is at most $2\rho_{\mathsf{CD}} = O(\rho_{\mathsf{CD}})$. ■

Bibliography

[1] B. Schlotfeldt, D. Thakur, N. Atanasov, V. Kumar, and G. J. Pappas, "Anytime planning for decentralized multirobot active information gathering," IEEE Robotics and Automation Letters, vol. 3, no. 2, pp. 1025–1032, 2018.

[2] N. Atanasov, J. Le Ny, K. Daniilidis, and G. J. Pappas, "Information acquisition with sensing robots: Algorithms and error bounds," in 2014 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2014, pp. 6447–6454.

[3] B. Charrow, V. Kumar, and N. Michael, "Approximate representations for multi-robot control policies that maximize mutual information," Autonomous Robots, vol. 37, no. 4, pp. 383–400, 2014.

[4] N. Atanasov, J. Le Ny, K. Daniilidis, and G. J. Pappas, "Decentralized active information acquisition: Theory and application to multi-robot slam," in IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 4775–4782.

[5] M. Corah and N. Michael, "Distributed matroid-constrained submodular maximization for multi-robot exploration: Theory and practice," Autonomous Robots, vol. 43, no. 2, pp. 485–501, 2019.

[6] D. Thakur, G. Loianno, W. Liu, and V. Kumar, "Nuclear environments inspection with micro aerial vehicles: Algorithms and experiments," in International Symposium on Experimental Robotics. Springer, 2018, pp. 191–200.

[7] A. Hilal, "An intelligent sensor management framework for pervasive surveillance," 2013.

[8] S. L. Smith, M. Schwager, and D. Rus, "Persistent robotic tasks: Monitoring and sweeping in changing environments," IEEE Transactions on Robotics, vol. 28, no. 2, pp. 410–426, 2011.

[9] V. Kumar, D. Rus, and S. Singh, "Robot and sensor networks for first responders," IEEE Pervasive computing, vol. 3, no. 4, pp. 24–33, 2004.

[10] R. Bajcsy, "Active perception," Proceedings of the IEEE, vol. 76, no. 8, pp. 966–1005, 1988.

[11] S. Thrun, W. Burgard, and D. Fox, Probabilistic robotics. MIT press, 2005.

[12] D. Fox, W. Burgard, and S. Thrun, "Active markov localization for mobile robots," Robotics and Autonomous Systems, vol. 25, no. 3-4, pp. 195–207, 1998.

[13] W. Burgard, D. Fox, and S. Thrun, "Active mobile robot localization," in IJCAI, 1997, pp. 1346–1352.

[14] J. Le Ny and G. J. Pappas, "On trajectory optimization for active sensing in gaussian process models," in Decision and Control, Proceedings of the 48th IEEE Conference on. IEEE, 2009, pp. 6286–6292.

[15] N. Atanasov, J. Le Ny, K. Daniilidis, and G. J. Pappas, "Information acquisition with sensing robots: Algorithms and error bounds," in Proceedings - IEEE International Conference on Robotics and Automation, 2014, pp. 6447–6454.

[16] B. Charrow, S. Liu, V. Kumar, and N. Michael, "Information-theoretic mapping using cauchy-schwarz quadratic mutual information," in 2015 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2015, pp. 4791–4798.

[17] A. Krause, "Optimizing sensing: Theory and applications," Ph.D. dissertation, Carnegie Mellon University, 2008.

[18] J. L. Williams, "Information theoretic sensor management," Ph.D. dissertation, Massachusetts Institute of Technology, 2007.

[19] V. Tzoumas, A. Jadbabaie, and G. J. Pappas, "Sensor placement for optimal kalman filtering: Fundamental limits, submodularity, and algorithms," in American Control Conference (ACC), 2016. IEEE, 2016, pp. 191–196.

[20] ——, "Near-optimal sensor scheduling for batch state estimation: Complexity, algorithms, and limits," in Decision and Control (CDC), 2016 IEEE 55th Conference on. IEEE, 2016, pp. 2695–2702.

[21] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—i," Mathematical programming, vol. 14, no. 1, pp. 265–294, 1978.

[22] B. Schlotfeldt, N. Atanasov, and G. J. Pappas, "Maximum information bounds for planning active sensing trajectories," in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2019, pp. 4913–4920.

[23] G. A. Hollinger and G. S. Sukhatme, "Sampling-based robotic information gathering algorithms," The International Journal of Robotics Research, vol. 33, no. 9, pp. 1271–1287, 2014.

[24] Y. Kantaros, B. Schlotfeldt, N. Atanasov, and G. J. Pappas, "Asymptotically optimal planning for non-myopic multi-robot information gathering." in Robotics: Science and Systems, 2019.

[25] M. Schwager, P. Dames, D. Rus, and V. Kumar, "A multi-robot control policy for information gathering in the presence of unknown hazards," in Robotics research. Springer, 2017, pp. 455–472.

[26] L. Zhou, V. Tzoumas, G. J. Pappas, and P. Tokekar, "Distributed attack-robust submodular maximization for multi-robot planning," arXiv preprint arXiv:1910.01208, 2019.

[27] W. Tabib, K. Goel, J. Yao, M. Dabhi, C. Boirum, and N. Michael, "Real-time information-theoretic exploration with gaussian mixture model maps."

[28] G. M. Hoffmann and C. J. Tomlin, "Mobile sensor network control using mutual information methods and particle filters," IEEE Transactions on Automatic Control, vol. 55, no. 1, pp. 32–47, 2009.

[29] K. Saulnier, D. Saldana, A. Prorok, G. J. Pappas, and V. Kumar, "Resilient flocking for mobile robot teams," IEEE Robotics and Automation letters, vol. 2, no. 2, pp. 1039–1046, 2017.

[30] P. Tokekar, E. Branson, J. Vander Hook, and V. Isler, "Tracking aquatic invaders: Autonomous robots for monitoring invasive fish," IEEE Robotics & Automation Magazine, vol. 20, no. 3, pp. 33–41, 2013.

[31] H.-L. Choi, "Adaptive sampling and forecasting with mobile sensor networks," Ph.D. dissertation, Massachusetts Institute of Technology, 2009.

[32] N. A. Atanasov, J. Le Ny, and G. J. Pappas, "Distributed algorithms for stochastic source seeking with mobile robot networks," Journal of Dynamic Systems, Measurement, and Control, vol. 137, no. 3, p. 031004, 2015.

[33] R. Sim and N. Roy, "Global a-optimal robot exploration in slam," in Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on. IEEE, 2005, pp. 661–666.

[34] L. Carlone, J. Du, M. K. Ng, B. Bona, and M. Indri, "Active slam and exploration with particle filters using kullback-leibler divergence," Journal of Intelligent & Robotic Systems, vol. 75, no. 2, pp. 291–311, 2014.

[35] M. Kontitsis, E. A. Theodorou, and E. Todorov, "Multi-robot active slam with relative entropy optimization," in American Control Conference (ACC), 2013. IEEE, 2013, pp. 2757–2764.

[36] T. H. Chung, J. W. Burdick, and R. M. Murray, "A decentralized motion coordination strategy for dynamic target tracking," in Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on. IEEE, 2006, pp. 2416–2422.

[37] C. M. Kreucher, "An information-based approach to sensor resource allocation," Ph.D. dissertation, University of Michigan, 2005.

[38] P. Dames and V. Kumar, "Autonomous localization of an unknown number of targets without data association using teams of mobile sensors," IEEE Transactions on Automation Science and Engineering, vol. 12, no. 3, pp. 850–864, 2015.

[39] B. J. Julian, M. Angermann, M. Schwager, and D. Rus, "Distributed robotic sensor networks: An information-theoretic approach," The International Journal of Robotics Research, vol. 31, no. 10, pp. 1134–1154, 2012.

[40] P. Dames, M. Schwager, V. Kumar, and D. Rus, "A decentralized control policy for adaptive information gathering in hazardous environments," in Decision and Control (CDC), 2012 IEEE 51st Annual Conference on. IEEE, 2012, pp. 2807–2813.

[41] G. M. Hoffmann and C. J. Tomlin, "Mobile sensor network control using mutual information methods and particle filters," IEEE Transactions on Automatic Control, vol. 55, no. 1, pp. 32–47, 2010.

[42] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, "Efficient informative sensing using multiple robots," Journal of Artificial Intelligence Research, vol. 34, pp. 707–755, 2009.

[43] F. Meyer, H. Wymeersch, M. Fröhle, and F. Hlawatsch, "Distributed estimation with information-seeking control in agent networks," IEEE Journal on Selected Areas in Communications, vol. 33, no. 11, pp. 2439–2456, 2015.

[44] M. Lauri and R. Ritala, "Stochastic control for maximizing mutual information in active sensing," in IEEE Int. Conf. on Robotics and Automation (ICRA) Workshop on Robots in Homes and Industry., 2014.

[45] S. Choudhury, A. Kapoor, G. Ranade, and D. Dey, "Learning to gather information via imitation," in Robotics and Automation (ICRA), 2017 IEEE International Conference on. IEEE, 2017, pp. 908–915.

[46] Y. V. Pant, H. Abbas, K. Mohta, T. X. Nghiem, J. Devietti, and R. Mangharam, "Co-design of anytime computation and robust control," in Real-Time Systems Symposium, 2015 IEEE. IEEE, 2015, pp. 43–52.

[47] M. Likhachev, G. J. Gordon, and S. Thrun, "Ara*: Anytime a* with provable bounds on sub-optimality," in Advances in Neural Information Processing Systems, 2004, pp. 767–774.

[48] N. Atanasov, J. Le Ny, K. Daniilidis, and G. J. Pappas, "Decentralized active information acquisition: Theory and application to multi-robot SLAM," 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 4775–4782, 2015.

[49] M. P. Vitus, W. Zhang, A. Abate, J. Hu, and C. J. Tomlin, "On efficient sensor scheduling for linear dynamical systems," Automatica, vol. 48, no. 10, pp. 2482–2493, 2012.

[50] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," IEEE transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100–107, 1968.

[51] L. Doyen, G. Frehse, G. J. Pappas, and A. Platzer, "Verification of hybrid systems," in Handbook of Model Checking. Springer, 2018, pp. 1047–1110.

[52] E. López, S. García, R. Barea, L. M. Bergasa, E. J. Molinos, R. Arroyo, E. Romera, and S. Pardo, "A multi-sensorial simultaneous localization and mapping (slam) system for low-cost micro aerial vehicles in gps-denied environments," Sensors, vol. 17, no. 4, p. 802, 2017.

[53] Y. Tian, K. Liu, K. Ok, L. Tran, D. Allen, N. Roy, and J. P. How, "Search and rescue under the forest canopy using multiple uas," in International Symposium on Experimental Robotics. Springer, 2018, pp. 140–152.

[54] J. P. Queralta, J. Taipalmaa, B. C. Pullinen, V. K. Sarker, T. N. Gia, H. Tenhunen, M. Gabbouj, J. Raitoharju, and T. Westerlund, "Collaborative multi-robot systems for search and rescue: Coordination and perception," arXiv preprint arXiv:2008.12610, 2020.

[55] F. Shkurti, A. Xu, M. Meghjani, J. C. G. Higuera, Y. Girdhar, P. Giguere, B. B. Dey, J. Li, A. Kalmbach, C. Prahacs et al., "Multi-domain monitoring of marine environments using a heterogeneous robot team," in 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2012, pp. 1747–1753.

[56] X. Lan and M. Schwager, "Rapidly exploring random cycles: Persistent estimation of spatiotemporal fields with multiple sensing robots," IEEE Transactions on Robotics, vol. 32, no. 5, pp. 1230–1244, 2016.

[57] G. Notomista and M. Egerstedt, "Persistification of robotic tasks," IEEE Transactions on Control Systems Technology, 2020.

[58] M. Popović, T. Vidal-Calleja, G. Hitz, J. J. Chung, I. Sa, R. Siegwart, and J. Nieto, "An informative path planning framework for uav-based terrain monitoring," Autonomous Robots, pp. 1–23, 2020.

[59] B. Grocholsky, J. Keller, V. Kumar, and G. Pappas, "Cooperative air and ground surveillance," IEEE Robotics & Automation Magazine, vol. 13, no. 3, pp. 16–25, 2006.

[60] G. S. Sukhatme, A. Dhariwal, B. Zhang, C. Oberg, B. Stauffer, and D. A. Caron, "Design and development of a wireless robotic networked aquatic microbial observing system," Environmental Engineering Science, vol. 24, no. 2, pp. 205–215, 2007.

[61] P. Tokekar, J. Vander Hook, D. Mulla, and V. Isler, "Sensor planning for a symbiotic uav and ugv system for precision agriculture," IEEE Transactions on Robotics, vol. 32, no. 6, pp. 1498–1511, 2016.

[62] Y. Rizk, M. Awad, and E. W. Tunstel, "Cooperative heterogeneous multi-robot systems: a survey," ACM Computing Surveys (CSUR), vol. 52, no. 2, pp. 1–31, 2019.

[63] S. Jorgensen, R. H. Chen, M. B. Milam, and M. Pavone, "The team surviving orienteers problem: routing teams of robots in uncertain environments with survival constraints," Autonomous Robots, vol. 42, no. 4, pp. 927–952, 2018.

[64] A. Viseras, Z. Xu, and L. Merino, "Distributed multi-robot information gathering under spatio-temporal inter-robot constraints," Sensors, vol. 20, no. 2, p. 484, 2020.

[65] D. Levine, B. Luders, and J. How, "Information-rich path planning with general constraints using rapidly-exploring random trees," in AIAA Infotech@ Aerospace 2010, p. 3360.

[66] N. A. Atanasov, "Active information acquisition with mobile robots," Ph.D. dissertation, University of Pennsylvania, 2015.

[67] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey, "An analysis of approximations for maximizing submodular set functions—ii," in Polyhedral combinatorics. Springer, 1978, pp. 73–87.

[68] N. Atanasov, R. Tron, V. M. Preciado, and G. J. Pappas, "Joint estimation and localization in sensor networks," in Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on. IEEE, 2014, pp. 6875–6882.

[69] R. Olfati-Saber, "Distributed kalman filtering for sensor networks," in Decision and Control, 2007 46th IEEE Conference on. IEEE, 2007, pp. 5492–5498.

[70] ——, "Kalman-consensus filter: Optimality, stability, and performance," in Decision and Control, Proceedings of the 48th IEEE Conference on. IEEE, 2009, pp. 7036–7042.

[71] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in Proceedings of the 4th international symposium on Information processing in sensor networks. IEEE Press, 2005, p. 9.

[72] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in Proceedings of the 28th International Conference on machine learning (ICML-11), 2011, pp. 465–472.

[73] F. Berkenkamp, M. Turchetta, A. P. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in Neural Information Processing Systems (NIPS), December 2017.

[74] U. Halder, B. Schlotfeldt, and P. Krishnaprasad, "Steering for beacon pursuit under limited sensing," in Decision and Control (CDC), 2016 IEEE 55th Conference on. IEEE, 2016, pp. 3848–3855.

[75] B. Yamauchi, "A frontier-based approach for autonomous exploration," in Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on. IEEE, 1997, pp. 146–151.

[76] N. Michael, J. Fink, and V. Kumar, "Experimental testbed for large multirobot teams," IEEE Robotics Automation Magazine, vol. 15, no. 1, pp. 53–61, March 2008.

[77] J. Guzzi, A. Giusti, L. M. Gambardella, G. Theraulaz, and G. A. D. Caro, "Human-friendly robot navigation in dynamic environments," in 2013 IEEE International Conference on Robotics and Automation, May 2013, pp. 423–430.

[78] J. Lee, V. S. Mirrokni, V. Nagarajan, and M. Sviridenko, "Non-monotone submodular maximization under matroid and knapsack constraints," in Proceedings of the forty-first annual ACM symposium on Theory of computing, 2009, pp. 323–332.

[79] X. Lan and M. Schwager, "Planning periodic persistent monitoring trajectories for sensing robots in gaussian random fields," in 2013 IEEE International Conference on Robotics and Automation. IEEE, 2013, pp. 2415–2420.

[80] M. Minoux, "Accelerated greedy algorithms for maximizing submodular set functions," in Optimization techniques. Springer, 1978, pp. 234–243.

[81] A. Mitra, J. A. Richards, S. Bagchi, and S. Sundaram, "Resilient distributed state estimation with mobile agents: overcoming byzantine adversaries, communication

losses, and intermittent measurements," Autonomous Robots, vol. 43, no. 3, pp. 743–768, 2019.

[82] A. Prorok, "Robust assignment using redundant robots on transport networks with uncertain travel time," IEEE Transactions on Automation Science and Engineering, 2020.

[83] A. Krause and D. Golovin, "Submodular function maximization," Tractability: Practical Approaches to Hard Problems, vol. 3, p. 19, 2012.

[84] J. B. Orlin, A. S. Schulz, and R. Udwani, "Robust monotone submodular function maximization," arXiv preprint:1507.06616, 2015.

[85] V. Tzoumas, K. Gatsis, A. Jadbabaie, and G. J. Pappas, "Resilient monotone submodular function maximization," in IEEE Conference on Decision and Control, 2017, pp. 1362–1367.

[86] V. Tzoumas, A. Jadbabaie, and G. J. Pappas, "Resilient non-submodular maximization over matroid constraints," arXiv preprint arXiv:1804.01013, 2018.

[87] M. Conforti and G. Cornuéjols, "Submodular set functions, matroids and the greedy algorithm," Discrete Applied Mathematics, vol. 7, no. 3, pp. 251 – 274, 1984.

[88] M. Sviridenko, J. Vondrák, and J. Ward, "Optimal approximation for submodular and supermodular optimization with bounded curvature," Math. of Operations Research, vol. 42, no. 4, pp. 1197–1218, 2017.

[89] F. Borrelli, A. Bemporad, and M. Morari, Predictive control for linear and hybrid systems. Cambridge University Press, 2017.

[90] U. Feige, "A threshold of $ln(n)$ for approximating set cover," Journal of the ACM, vol. 45, no. 4, pp. 634–652, 1998.

[91] G. Nemhauser, L. Wolsey, and M. Fisher, "An analysis of approximations for maximizing submodular set functions – I," Mathematical Programming, vol. 14, no. 1, pp. 265–294, 1978.

[92] D. Sharma, A. Kapoor, and A. Deshpande, "On greedy maximization of entropy," in Proceedings of the 32nd International Conference on Machine Learning, 2015, pp. 1330–1338.

[93] V. Tzoumas, N. A. Atanasov, A. Jadbabaie, and G. J. Pappas, "Scheduling nonlinear sensors for stochastic process estimation," in American Control Conference, 2017, pp. 580–585.

[94] L. F. Chamon, G. J. Pappas, and A. Ribeiro, "The mean square error in kalman filtering sensor selection is approximately supermodular," in IEEE 56th Annual Conference on Decision and Control (CDC), 2017, pp. 343–350.

[95] J. B. Orlin, A. S. Schulz, and R. Udwani, "Robust monotone submodular function maximization," in Inter. Conference on Integer Programming and Combinatorial Optimization, 2016, pp. 312–324.

[96] S. T. Jawaid and S. L. Smith, "Submodularity and greedy algorithms in sensor scheduling for linear dynamical systems," Automatica, vol. 61, pp. 282–288, 2015.

[97] S. Thrun, "Probabilistic robotics," Communications of the ACM, vol. 45, no. 3, pp. 52–57, 2002.

[98] N. Michael, E. Stump, and K. Mohta, "Persistent surveillance with a team of mavs," in 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2011, pp. 2708–2714.

[99] H. Jeong, B. Schlotfeldt, H. Hassani, M. Morari, D. D. Lee, and G. J. Pappas, "Learning q-network for active information acquisition," arXiv preprint arXiv:1910.10754, 2019.

[100] V. Tzoumas, A. Jadbabaie, and G. J. Pappas, "Resilient Monotone Sequential Maximization," ArXiv e-prints: 1803.07954.